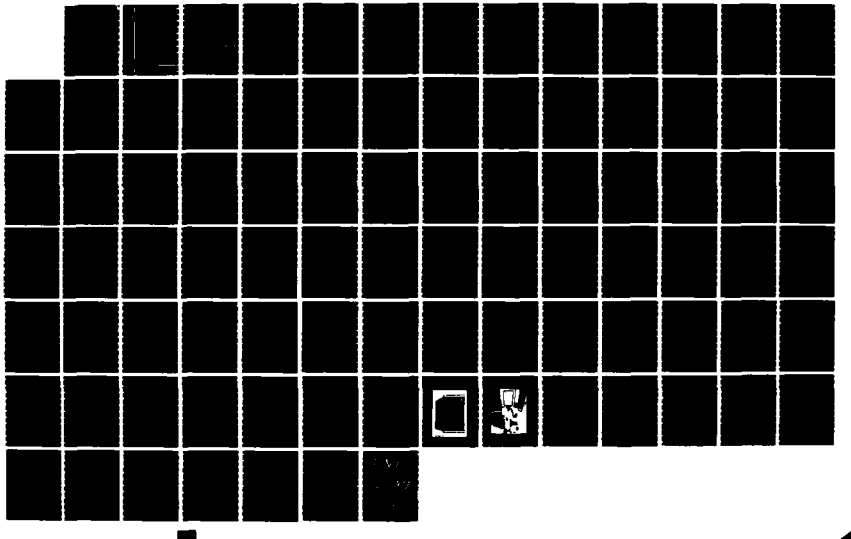
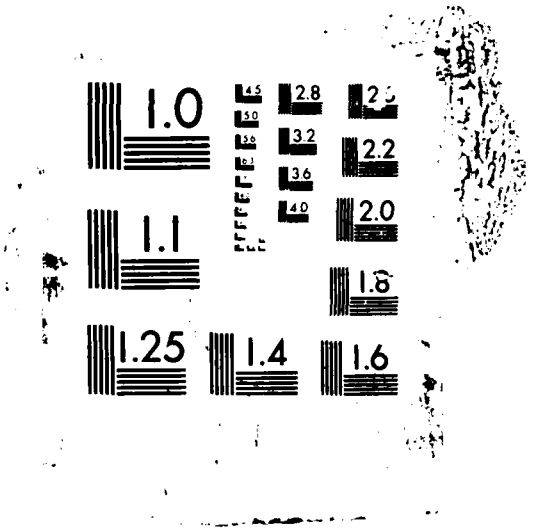
AD-A176 039 IMPLEMENTATION OF HIGH SPEED FFTS FOR RADAR SIGNAL PROCESSING(U) COMMUNICATIONS RESEARCH CENTRE OTTAWA (ONTARIO) H C CHAN AUG 85 CRC-1394 1/1

UNCLASSIFIED F/G 17/9 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.2
2.0
1.8

Communications Research Centre

IMPLEMENTATION OF HIGH SPEED FFTs FOR RADAR SIGNAL PROCESSING

by

H.C. Chan

This work was sponsored by the Department of National Defence, Research and Development Branch under Sub Program 33C.

CRC REPORT NO. 1394
OTTAWA, AUGUST 1985

Government of Canada
Department of Communications

Gouvernement du Canada
Ministère des Communications

Canada

COMMUNICATIONS RESEARCH CENTRE

DEPARTMENT OF COMMUNICATIONS
CANADA

IMLEMENTATION OF HIGH SPEED FFTs FOR RADAR SIGNAL PROCESSING

by

H.C. Chan

(Radar and Communications Technology Branch)

CRC REPORT NO. 1394

August 1985
OTTAWA

This work was sponsored by the Department of National Defence,
Research and Development Branch under Sub Program 33C.

TABLE OF CONTENTS PAGE

LIST OF FIGURES iii

LIST OF TABLES iv
ABSTRACT 1

1. INTRODUCTION 1

1.1 A survey of existing hardware FFT processors 2

2. THE FFT ALGORITHM 3

2.1 Two-dimensional representation of the DFT of a one-dimensional sequence 3
2.2 Generalized FFT procedures 7
2.2.1 Twiddle factor 8
2.2.2 Butterfly 10
2.2.3 Radix 12
2.3 Alternative decimation processes 12
2.3.1 Decimation-in-frequency 12
2.3.2 Decimation-in-time 13
2.3.3 Digit reversal 14

3. REDUCING THE COMPUTATIONAL REQUIREMENTS FOR DFT ALGORITHM AND ITS PROCESSING COMPONENTS 15

3.1 Reduction of computational effort 15
3.2 Radix-r butterfly and radix-r FFT 16
3.2.1 Radix-2 butterfly and 2-point DFT 18
3.2.2 Radix-4 butterfly and 4-point DFT 20
3.3 Pipelining of the butterfly for maximum throughput 23

4. HARDWARE IMPLEMENTATION OF THE FFT ALGORITHM 27

4.1 Parallel-pipeline FFT 29
4.1.1 Radix-2 parallel-pipeline 16-point FFT 29
4.1.2 Radix-4 parallel-pipeline 16-point FFT 30
4.2 Single-pipeline FFTs 30
4.2.1 Radix-2 single-pipeline 16-point FFT 32
4.2.2 Radix-4 single-pipeline 16-point FFT 40
4.3 Radix-r single pipeline FFT 49

5. DESIGN FOR A DOPPLER PROCESSOR DISCRIMINATOR AND A 2-DIMENSIONAL DIGITAL BEAMFORMER BASED ON A HIGH SPEED 16-POINT FFT IMPLEMENTATION 51

5.1 Doppler processing and digital beamforming 51

5.2 Implementation of a 2-Dimensional digital beamformer and a Target Doppler processor 61
5.2.1 A modified 2-dimensional digital beamformer 61
a. modified parallel-pipeline 16-point FFT 61
b. An 8x16 row-column transposition network 63
5.2.2 Target Doppler Processor employing the modified parallel-pipeline 16-point FFT 64

6. HIGH SPEED FFT PROCESSORS FOR OTHER RADAR APPLICATIONS 68

6.1 Digital pulse compression and radar image processing 68
6.2 Component count reduction versus throughput reduction 73
6.3 Parallel-pipeline FFTs of larger dimension 75

7. ACKNOWLEDGEMENT 77

8. REFERENCES 78

LIST OF FIGURES PAGE

Figure 1 Signal flow diagram of the generalized FFT procedures 11
Figure 2 Symbolic diagram of a radix-2 butterfly and schematic diagram of a 2-point DFT 19
Figure 3 Symbolic diagram of a radix-4 butterfly 21
Figure 4 Schematic diagram of a 4-point DFT 22
Figure 5 Schematic diagram of a pipelined radix-2 butterfly 25
Figure 6 Schematic diagram of a pipelined 4-point DFT 26
Figure 7 Signal flow diagram of the radix-2 DIT 16-point FFT 28
Figure 8 Signal flow diagram of the radix-4 16-point FFT 31
Figure 9 Signal flow diagram of the radix-2 DIF 16-point FFT 33
Figure 10 Data buffering scheme for a radix-2 butterfly 34
Figure 11 Schematic diagram of a radix-2 commutator 35
Figure 12 Summary of steps of the generalized FFT procedures as applied to the radix-2 DIF 16-point case 37
Figure 13 Schematic diagram of a radix-2 DIF single-pipeline FFT ... 39
Figure 14 Data buffering scheme for a radix-4 butterfly 41
Figure 15 Schematic diagram of a radix-4 commutator 42
Figure 16 Illustration of the input-output data arrangement for a radix-4 commutator 45
Figure 17 Summary of steps of the generalized FFT procedures as applied to the radix-4 16-point case 46
Figure 18 Schematic diagram of a radix-4 single-pipeline 16-point FFT 47
Figure 19 Schematic diagram of a radix-r single-pipeline FFT 50
Figure 20 Digital representation of radar signals 52
Figure 21 Block diagram of a radar signal processing system employing a target Doppler discriminator 55
Figure 22 System block diagram of an 8x8 SAMPAR system 58
Figure 23 Data permutation in a 2-dimensional beamformer 59
Figure 24 Schematic diagram of a 16x16 2-Dimensional digital beamformer 60
Figure 25 Schematic diagram of a modified parallel-pipeline 16-point FFT processor 62
Figure 26 Schematic diagram of a 8x16 row-column transposition network 65
Figure 27 Required data arrangement for the 8x16 row-column transposition network 66
Figure 28 Operation of the 8x16 row-column transposition network at various clock cycles 67
Figure 29 Schematic diagram of a Target Doppler discriminator employing a modified parallel-pipeline 16-point FFT 69
Figure 30 Prototype of the Doppler Processor 70
Figure 31 Prototype of the 2-D digital Beamformer 71

LIST OF TABLES PAGE

Table I. Two-dimensional representation of the DFT of a one-dimensional sequence 7
Table II. Two-dimensional representation of a one-dimensional sequence 8
Table III. Two-dimensional representation of the twiddle factor 9
Table IV. Result of column-DFTs and twiddle multiplication .. 9
Table V. Steps of the generalized FFT procedures 10
Table VI. Data arrangement of a time sequence using the decimation-in-frequency procedure 12
Table VII. Data arrangement of the DFT of a time sequence using the decimation-in-frequency procedure 13
Table VIII. Data arrangement of a time sequence using the decimation-in-time procedure 13
Table IX. Required number of complex multiplications for an N-point radix-2 and radix-4 FFTs 23
Table X. Component counts of pipelined 2-point and 4-point DFTs and twiddle multiplier 27
Table XI. Component count for a radix-2 parallel-pipeline 16-point FFT 30
Table XII. Component count for a radix-4 parallel-pipeline 16-point FFT 30
Table XIII. Component count for a radix-2 single-pipeline 16-point FFT 40
Table XIV. Switch connections as a function of clock cycles for the MUXs in a radix-4 commutator 43
Table XV. Component count for a radix-4 single-pipeline 16-point FFT 48
Table XVI. Comparison of component counts for various 16-point pipeline FFT structures 48
Table XVII. Comparison of component counts for various 64-point pipeline FFT structures 49
Table XVIII. Component count of a modified parallel-pipeline 16-point FFT 63
Table XIX Switch connections of the MUXs in the 8x16 row-column transposition network through 20 clock cycles 64
Table XX. Comparison of IC chip counts between two implementations of the 2-D digital beamformer 74

IMPLEMENTATION OF HIGH SPEED FFTs FOR RADAR SIGNAL PROCESSING

by

H.C. Chan

ABSTRACT

Discrete Fourier Transform (DFT) has found wide application in radar and sonar systems. With an increased emphasis on digital signal processing, radar systems have requirements for DFTs with ever increasing data rates. The Fast Fourier Transform (FFT) algorithm permits efficient computation of the DFT. In this work, both the fundamentals of FFT algorithms and their implementation are discussed, with emphasis on hardware design. Designs are presented for two radar signal processors employing high speed FFTs, i.e., (1) a high speed Doppler processor and (2) a two-dimensional digital beam-former.

1. INTRODUCTION

The discrete Fourier Transform (DFT) has found wide application in radar and sonar systems. In the case of radar, there is a requirement for DFT processors capable of very high data rates. Some examples of typical radar applications are the following: target Doppler processing, pulse Doppler processing, pulse compression, matched filtering, radar imagery processing and, recently, digital beam-forming. The processing speed requirements range from several MHz in the case of a Doppler processor to over a GHz for a digital beam-former. In this report, some aspects of the hardware implementation of the Fast Fourier Transform (FFT) are considered. It is intended in this report to provide some insight into the problem of developing:

(a) techniques for implementing the FFT algorithm in a form suitable for radar signal processing,

(b) techniques for increasing processing speed and efficiency,

(c) techniques for reducing the hardware component count,

(d) techniques for achieving greater parallelism so that higher throughput rate can be realized.

1.1 A survey of existing hardware FFT processors

It is instructive to survey the literature and determine if any "off-the-shelf" hardware FFT processors are able to fulfill the requirements outlined in the previous section. In a paper published in 1969, Bergland [1] tabulated the characteristics of over twenty hardware FFT processors which existed at that time. The comparison is based on processor architecture, function and performance characteristics, system hardware features and cost. Most of these processors were produced by research laboratories, such as Bell Laboratory, Stanford Research Institute, MIT Lincoln Laboratory, for specific applications. A few were produced by computer manufacturers such as IBM, Control Data Corp. Processing speeds for these processors were measured in terms of the time required to compute a 1024-point complex FFT. The speeds range from a high of 1 ms for the MM DSP of Emerson Electric to a low of 600 ms for the CSS-3 of Computer Signal Processors, Inc. Typically, the execution time for a 1024-point complex FFT is in the tens of milliseconds. These times compare very favorably with those achieved by present-day commercial dat- -cessors.

Over a decade since the publication of this survey paper. Many special purpose .- _.. peed FFT processors undoubtedly have been built by various research laboratories and in the industry. However, their technical characteristics are not generally available. More importantly, there appears to be no commercially available processors which are directly applicable to real-time radar signal processing functions. Commercially available FFT processors usually take the form of an array processor. Array processors lean towards high flexibility and tend to be software orientated. Examples of these processors include the Floating Point System (FPS) family of array processors, Star-100 array processor developed by Star Technologies Inc., and the TASP array processor developed by ESE. The throughput rates of these systems, measured in terms of millions of floating point operations (Mega-flops), are very impressive, ranging from 25 Mega-flops for an AP-120B to over 100 Mega-flops for the TASP. With respect to FFT processing, the execution time for a 1024-point complex FFT ranges from 5 msec for a FPS AP120B to about 800 micro-seconds for the TASP. However, the operation of array processors is input/output (I/O)-limited. For applications such as a 2-D digital beamformer, where the data are available simultaneously, their throughput rate is limited by the data transfer rate of the I/O interface.

Families of IC chips specially designed for FFT processing are available from a few device manufacturers. Examples of these IC chips are the AM29500 family by Advanced Micro Devices, and the Weitek family signal processing ICs. Recently, IBM and Sony of Japan each introduced a family of monolithic devices suitable for the implementation of systolic array processors[2]. These devices may be used to form the building blocks of a high speed FFT processor for real-time radar applications. However, the cost of these devices is currently very high. Consequently, it is more economical to develop high speed digital radar signal processors with inexpensive and commercially available ICs.

It is well known that there tends to be a tradeoff between processing speed and processing flexibility in the design of digital hardware. Due

to the extremely high data rate requirements in radar applications, it is usually necessary to sacrifice flexibility in favour of speed and to use special purpose hardware. In a later section, the designs of a target Doppler processor and a two-dimensional digital beamformer will be discussed. These two examples will serve to demonstrate how one maximizes throughput rate and minimizes component count.

2. THE FFT ALGORITHM

In order to select the most suitable approach to the hardware implementation of an algorithm, it is essential that one thoroughly understands the structure of the algorithm. This understanding will enable the circuit designer to take advantage of any parallelism or special constructs of the algorithm and match these to the capabilities of state-of-the-art hardware components.

In most standard digital signal processing texts [3]-[5], the FFT algorithm is explained both in terms of the decimation-in-time (DIT) and the decimation-in-frequency (DIF) processes. Generally, when these descriptions are applied to the FFT algorithm, they are meant to refer to any procedure in which the DFT of an N-point sequence is obtained by first computing the DFTs of two or more sub-sequences; i.e., by decimating the original sequence and then combining them in some fashion. The premise is that the total effort required for computing the smaller DFTs and combining the results is always less than that required for directly computing the DFT of the original N-point sequence. We will not proceed immediately with a proof of this premise, but rather, we will first discuss the basic concept underlying the FFT. It will then be shown that this decomposition reduces the computational overhead.

2.1 Two-dimensional representation of the DFT of a one-dimensional sequence

The DFT of an N-point complex sequence $S:\{x_0,x_1,x_2,..,x_{N-1}\}$ is defined as:

$$F_k = \sum_{n=0}^{n=N-1} x_n \exp(-j2\pi nk/N) \qquad (1)$$

$$k=0,1,2,\ldots,N-1$$

It is clear that to evaluate F_k for all N values of k, $(N-1)^2$ complex multiplications and N(N-1) complex additions are required. A complex multiplication requires four real multiplications and two real additions. A complex addition requires two real additions. In subsequent discussions, unless otherwise noted, the terms multiplication and addition are understood to mean complex multiplication and addition, respectively. This estimate of the number of multiplications takes into account the fact that the exponential term $\exp(-j2\pi nk/N)$ is unity for n=k=0. Although the term FFT was used initially in the work of Tukey et al [6] to mean a specific algorithm, much progress has been made since. Consequently, the FFT algorithm may now be regarded as any computational procedure which reduces significantly the required computational effort in evaluating Eqn (1).

It can be shown that both the so-called DIT and DIF processes lead to the same interpretation of the FFT algorithm. Rabiner and Gold[3] developed the mathematical treatment which led to a unified approach to the FFT. Their approach is based on the two-dimensional representation of a one-dimensional sequence. The DFT sum is transformed into a double summation. In the following, we shall present a development which shows how the two-dimensional representation of the DFT can be arrived at through the process of decimation. It will provide the linkage between the concepts of DIT and DIF. Some of the special terms such as butterfly, radix, twiddle factor and digit reversal will become clear as the development unfolds.

Consider an N-point complex sequence $S:\{x_0,x_1,x_2,..,x_{N-1}\}$. If N is not a prime number, then it can be expressed as a product of at least two numbers L and M, i.e.,

$$N = L \times M \qquad (2)$$

The N-point sequence can be expressed as the sum of a number of auxiliary sequences. These auxiliary sequences are constructed from the original sequence by retaining a subset of the samples and padding the rest with zeros. One particular construction is as follows:

The N-point sequence is expressed as the sum of L auxiliary sequences S^0, S^1, S^2,...,S^{L-1}, which are defined in Eqn (3).

$$\begin{aligned}
S^0 &: x_0,x_1,x_2,\ldots,x_{M-1},0,0,\ldots,0,\ldots,0,0,0,\ldots,0\\
S^1 &: 0,0,0,\ldots,x_M,x_{M+1},\ldots,x_{2M-1},0,\ldots,0,0,0,0,\ldots,0\\
&\vdots\\
S^{L-1} &: 0,0,0,\ldots,0,0,0,\ldots,0,\ldots,x_{(L-1)M},x_{(L-1)M+1},\ldots x_{N-1}
\end{aligned} \qquad (3)$$

M Entries M Entries M Entries

These auxiliary sequences contain M non-zero entries, and each is padded with (N-M) zeros. The non-zero entries in each auxiliary sequence is a subset of M contiguous samples taken from the original sequence. There is no overlapping of non-zero entries among the auxiliary sequences. Since the DFT is a linear operation, the DFT of the original N-point sequence is equal to the sum of the DFTs of the L auxiliary sequences.

Let the DFTs of the auxiliary sequences S^0, S^1, S^2,..., S^{L-1}, be F_k^0, F_k^1, ..., $F_k^{(L-1)}$, respectively. Since there are only M non-zero entries in each auxiliary sequence, the corresponding DFTs reduce to those defined in Eqn (4).

$$F_k^0 = \sum_{m=0}^{M-1} x_m \exp(-j\frac{2\pi mk}{LM})$$

$$F_k^1 = \sum_{m=0}^{M-1} x_{m+M} \exp(-j\frac{2\pi mk}{LM}) \exp(-j\frac{2\pi k}{L}) \qquad (4)$$

$$\cdot \quad \cdot$$

$$\cdot \quad \cdot$$

$$F_k^{(L-1)} = \sum_{m=0}^{M-1} x_{m+(L-1)M} \exp(-j\frac{2\pi mk}{LM}) \exp[-j\frac{2\pi(L-1)k}{L}]$$

$$k = 0,1,2,3,\ldots N-1$$

From Eqn (2) we have substituted L x M for N in Eqn (1). Using superposition, we can express the DFT of sequence S as:

$$F_k = F_k^{\,0} + F_k^{\,1} + F_k^{\,2} + \ldots F_k^{(L-1)}$$

$$= \sum_{m=0}^{M-1} x_m \exp(-j\frac{2\pi mk}{LM}) + \sum_{m=0}^{M-1} x_{m+M} \exp(-j\frac{2\pi mk}{LM}) \exp(-j\frac{2\pi k}{L})$$

$$+ \ldots + \sum_{m=0}^{M-1} x_{m+(L-1)M} \exp(-j\frac{2\pi mk}{LM}) \exp[-j\frac{2\pi(L-1)k}{L}] \qquad (5)$$

$$k = 0,1,2,\ldots,N-1$$

Equation (5) can be conveniently expressed by a double summation with the introduction of the index ℓ

$$F_k = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi mk}{LM}) \exp(-j\frac{2\pi \ell k}{L}) \qquad (6)$$

$$k = 0,1,2,\ldots,N-1$$

At this point the symmetrical properties of the DFT play an important role in the development of the FFT algorithm. Notice that the last exponential term of the double summation in Eqn (6) repeats in a cyclic fashion as ℓk, mod. L. Consider the following sub-groups of frequency samples;
$\{F_0, F_L, F_{2L}, \ldots F_{(M-1)L}\}$; $\{F_1, F_{1+L}, F_{1+2L}, \ldots F_{1+(M-1)L}\}$; $\ldots$;
$\{F_{L-1}, F_{(L-1)+L}, F_{(L-1)+2L}, \ldots F_{(L-1)+(M-1)L}\}$ separately. The first

sequence of frequencies can be expanded as:

$$F_0 = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M}$$

$$F_L = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi m}{M}) \tag{7}$$

$$\cdot \quad \cdot \quad \cdot$$

$$F_{(M-1)L} = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp[-j\frac{2\pi m(M-1)}{M}]$$

With the introduction of index r, these equations can be expressed as the following set of equations:

$$F_{rL} = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi mr}{M}) \tag{8}$$

$$r = 0,1.2.,,,.M-1$$

Similarly, for the next set, we have:

$$F_1 = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi m}{LM}) \exp(-j\frac{2\pi \ell}{L}) \tag{9}$$

$$F_{1+L} = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi m}{LM}) \exp(-j\frac{2\pi m}{M}) \exp(-j\frac{2\pi \ell}{L})$$

$$\cdot \quad \cdot \quad \cdot$$

$$F_{1+(M-1)L} = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi m}{LM}) \exp[-j\frac{2\pi m(M-1)}{M}] \exp(-j\frac{2\pi \ell}{L})$$

which can be expressed as a sequence with index r:

$$F_{1+rL} = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi m}{LM}) \exp(-j\frac{2\pi mr}{M}) \exp -j\frac{2\pi \ell}{L}) \tag{10}$$

$$r=0,1,2,\ldots,M-1$$

A pattern begins to emerge. We see that the DFT can now be regarded as L sequences each of which contains M entries. Consequently, we can express the DFT of the original N-point sequence S as a two-dimensional

array by introducing the index s:

$$F_k = F_{s+rL} = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M-1} x_{m+\ell M} \exp(-j\frac{2\pi ms}{LM}) \exp(-j\frac{2\pi mr}{M}) \exp(-j\frac{2\pi \ell s}{L}) \qquad (11)$$

$$s = 0,1,2,\ldots,L-1$$
$$r = 0,1,2,\ldots,M-1$$

Equation (11) represents the DFT of the sequence S: $\{x_0, x_1, \ldots x_{N-1}\}$, for all values of k=0,1,2,3,...,N-1. Graphically, the frequency components of the DFT can be seen as being arranged into a two-dimensional array as shown in Table I.

Table I: Two-dimensional representation of the DFT of a one-dimensional sequence.

s \ r	0	1	2	.....	M-1
0	F_0	F_L	F_{2L}	.	$F_{(M-1)L}$
1	F_1	F_{1+L}	F_{1+2L}	.	$F_{1+(M-1)L}$
2	F_2	F_{2+L}	F_{2+2L}	.	$F_{2+(M-1)L}$
3	F_3	F_{3+L}	F_{3+2L}	:	$F_{3+(M-1)L}$
.	.	.	.	.	.
.	.	.	.	.	.
L-1	F_{L-1}	$F_{(L-1)+L}$	$F_{(L-1)+2L}$	.....	$F_{(L-1)+(M-1)L}$

2.2 Generalized FFT procedures

We now turn our attention to the double summation in Eqn (11). After rearranging terms, we obtain:

$$F_{s+rL} = \sum_{m=0}^{M-1} G_{ms} \exp(-j\frac{2\pi mr}{M}) \qquad (12)$$

$$s = 0,1,2,\ldots,L-1$$
$$r = 0,1,2,\ldots,M-1$$

where

$$G_{ms} = \exp(-j\frac{2\pi ms}{LM}) \sum_{\ell=0}^{L-1} x_{m+\ell M} \exp(-j\frac{2\pi \ell s}{L})$$

From the basic definition of the DFT, we recognize immediately that summation over ℓ in the term G_{ms} for a fixed m, is simply the DFT of a decimated sequence of L samples of the original N-point sequence S. To be precise, the decimated sequences are obtained by taking every other M samples from the original N-point sequence with the index, m, indicating the first sample to be taken. For example, for m=0, the decimated sequence will be: $\{x_0\ x_M,\ x_{2M},\ldots,x_{(L-1)M}\}$, and for m=1, the decimated sequence will be $\{x_1,\ x_{1+M},\ x_{1+2M},\ldots,\ x_{1+(L-1)M}\}$ and so on. It is easy to visualize this particular decimation process by dividing the original N-point sequence into L contiguous parts of M samples each and then stacking them on top of one another as shown in Table II. We see that the resulting columns are exactly the required decimated sequences.

Table II: Two dimensional representation of a one-dimensional sequence.

ℓ \ m	0	1	2	.	M-1
0	x_0	x_1	x_2	.	$x_{(M-1)}$
1	x_M	x_{1+M}	x_{2+M}	.	$x_{(M-1)+M}$
2	x_{2M}	x_{1+2M}	x_{2+2M}	.	$x_{(M-1)+3M}$
3	x_{3M}	x_{1+3M}	x_{2+3M}	. . . . . .	$x_{(M-1)+2M}$ . . . . .
L-1	$x_{(L-1)M}$	$x_{1+(L-1)M}$	$x_{2+(L-1)M}$	.....	$x_{(M-1)+(L-1)M}$

It becomes apparent that the summation over index ℓ in Eqn (12) for all m are precisely the DFTs of all the columns in Table II. After the DFTs of all columns in Table II are performed, the results will be in terms of both indices m and s, where s is the frequency index of the column-DFTs.

2.2.1 Twiddle factors

According to Eqn (12), the DFTs of all columns are to be multiplied by a factor $\exp(-2\pi ms/N)$. This factor is commonly known as the twiddle factor. Most authors avoid writing the exponential explicitly by defining the parameter W given by:

$$W = \exp(-j2\pi/N) \tag{13}$$

Consequently, the twiddle factor may be written as W^{ms}. Since the twiddle factor is a function of two variables, m and s, it can also be represented as a two-dimensional array, as shown in Table III.

Table III: Two-dimensional representation of the twiddle factor.

s \ m	0	1	2	.....	M-1
0	1	1	1	.....	1
1	1	$\exp(-j\frac{2\pi}{N})$	$\exp(-j\frac{2\pi 2}{N})$	.	$\exp[-j\frac{2\pi(M-1)}{N}]$
2	1	$\exp(-j\frac{2\pi 2}{N})$	.	.	.
3 . . .	1 . . .	. . . .	. . . .	. . . .	. . . .
L-1	1	$\exp[-j\frac{2\pi(L-1)}{N}]$	.	.	$\exp[-j\frac{2\pi(L-1)(M-1)}{N}]$

When W^{ms} is equal to unity or $\pm j$, no real multiplications are required. We shall call these three values of the twiddle factor trivial twiddles.

It should be pointed out that twiddle multiplication should not be confused with matrix multiplication. It is an operation where entries from corresponding locations of the DFT and twiddle matrices are multiplied and a new matrix formed from the product. Let us represent the result of the twiddle multiplication by a 2-dimensional array as shown in Table IV.

Table IV: Results of the twiddle multiplication with results of the column-DFTs

s \ m	0	1	2	. . . . .	M-1
0	$G_{0,0}$	$G_{1,0}$	$G_{2,0}$	. . . . .	$G_{M-1,0}$
1	$G_{0,1}$	$G_{1,1}$	$G_{2,1}$	. . . . .	$G_{M-1,1}$
2	$G_{0,2}$	$G_{1,2}$	$G_{2,2}$	. . . . .	$G_{M-1,2}$
. .	. .	. .	. .	. .	. .
L-1	$G_{0,L-1}$	.	.	.	$G_{(M-1),(L-1)}$

By referring both to Eqn (12) and Table IV, we see that the summation over m in Eqn (12) consists of the DFTs of all the rows in Table IV. The steps described above constitute the generalized FFT procedures. A variety of FFT algorithms can be derived by applying these decimation steps in different ways to the original sequence. The steps of the generalized FFT procedures are summarized in Table V.

Table V: Generalized FFT procedures

STEP	OPERATIONS
i.	The starting N-point sequence is divided into L sub-sequences of M contiguous samples, and each sub-sequence is used as a row to form an L by M matrix (N = LxM).
ii.	The elements of an L-by-M twiddle matrix are formed using the twiddle factor defined by: $W^{ms} = \exp(-j2\pi ms/N)$ s=0,1,2,3,..,L-1 m=0,1,2,..,M-1 where the row and column indices are given respectively by s and m.
iii.	L-point DFTs are performed on all M columns and the result placed in the identical locations of the data matrix.
iv.	The elements of the matrix formed in (iii) are multiplied by the twiddle matrix(element by element).
v.	Finally, M-point DFTs are performed on all L rows of the matrix resulting from step iv.

2.2.2 Butterfly

The generalized FFT procedures given in Table V can be conveniently represented graphically by a signal flow diagram using a special diagrammatic representation called the butterfly. Basically, a butterfly operation consists of a DFT whose inputs and outputs can be multiplied by a set of complex weights called twiddle factors. The signal-flow diagram of the generalized FFT procedures is shown in Figure 1a. The symbolic diagrams of the butterfly capable of processing r complex input samples are shown for the DIT and DIF processes in Figures 1b and 1c, respectively. These butterflies have r input-nodes and r output-nodes, where r is a factor of the length of the sequence N. The convention commonly adopted is that the butterfly will take time sequence data in natural order (from top to bottom) and produce frequency data at the output-nodes in the same order. The twiddle multiplier comprises one trivial and (r-1) non-trivial twiddle multiplications.

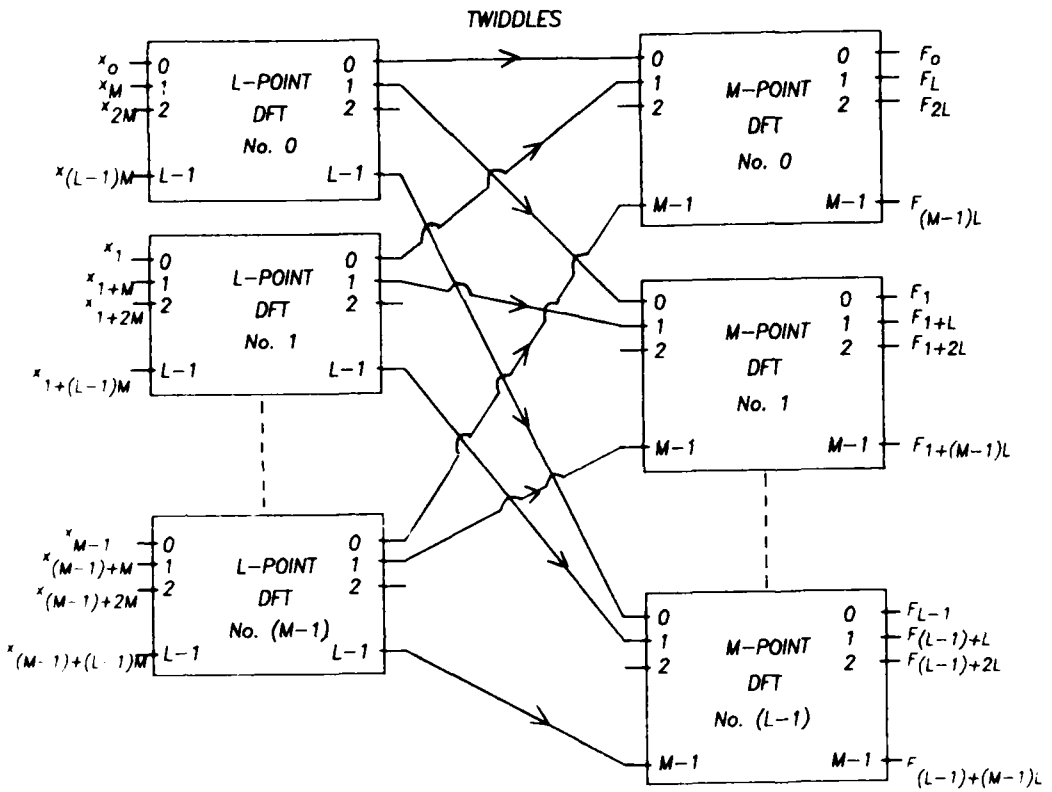
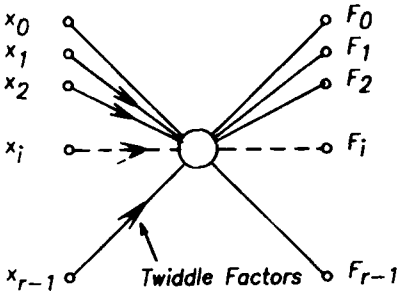
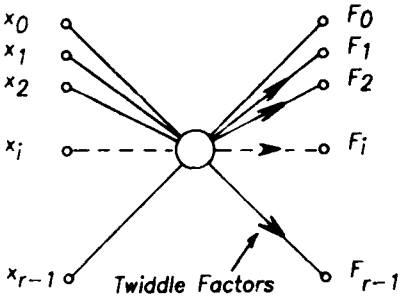


FIGURE 1. (a) SIGNAL FLOW DIAGRAM OF THE GENERALIZED FFT PROCEDURES.



(b) RADIX–r DIT BUTTERFLY.



(c) RADIX–r DIF BUTTERFLY.

2.2.3 Radix

The number of input or output samples processed by a butterfly is called the radix. We may consider the signal-flow diagram (Figure 1a) of the generalized FFT algorithm as being composed of two stages. The first stage comprises M radix-L butterflies, and the second stage comprises L radix-M butterflies. Notice that the twiddle factors may be considered as part of the first stage butterflies, in which case, the twiddles are used to post-multiply the output of the radix-L DFTs. Alternatively, the twiddle factors may be considered as part of the second stage butterflies. In this case, the twiddle factors are used to pre-multiply the input of the radix-M DFTs.

2.3 Alternative decimation processes

So far we have shown that if N is a product of two numbers, L and M, then by decomposing the sequence into L auxiliary sequences of M samples, the original DFT can be interpreted as a two dimensional array with L rows and M columns. Intuitively, by interchanging L and M, we should be able to treat the DFT of the N-point sequence as an array consisting of M rows and L columns. It can be shown that the two decimation processes produce results which are mathematically equivalent, and they correspond to the so-called decimation-in-frequency (DIF) and decimation-in-time (DIT) processes respectively.

Historically, the so-called DIF and DIT processes represent the two alternative ways of decimating an N-point sequence into two sequences: (i) DIF: representation as an (N/2)-column by 2-row matrix, and (ii) DIT: representation as a 2-column by (N/2)-row matrix.

2.3.1 Decimation-in-frequency

To carry out a decimation-in-frequency, let L=2, from which it follows that M=N/2. When the first step of the generalized FFT procedures given in Table V is applied to the N-point sequence, we obtain the matrix given in Table VI.

Table VI: Two-dimensional arrangement of time samples for the DIF process.

ℓ \ m	0	1	2	. . . . .	$\frac{N}{2} - 1$
0	x_0	x_1	x_2	. . . . .	$x_{\frac{N}{2}-1}$
1	$x_{\frac{N}{2}}$	$x_{\frac{N}{2}+1}$	$x_{\frac{N}{2}+2}$	. . . . .	x_{N-1}

With the application of 2-point DFTs to the columns of Table VI, followed by twiddle multiplications and DFTs applied to the rows, we obtain the results shown in Table VII.

Table VII: Two-dimensional arrangement of DFT samples for the DIF process.

s \ r	0	1	2	. . . . .	$\frac{N}{2} - 1$
0	F_0	F_2	F_4	. . . . .	F_{N-2}
1	F_1	F_3	F_5	. . . . .	F_{N-1}

Since the upper and lower rows in Table VII give the even and odd samples of the transform, respectively, the procedure used in deriving this results is called the decimation-in-frequency process.

In general, the DIF process refers to the decimation of a sequence using an r-row by (N/r)-column representation of the original sequence. In this case, the column-DFTs are of dimension r, and the twiddle multiplications are performed after the column-DFTs. Hence, these two operations (DFTs and twiddle multiplications) can be conveniently performed by radix-r butterflies in which the twiddles are applied after the r-point DFT. If N is expressible as a power of r, the row-DFTs can be further decomposed using the same generalized FFT procedures (i.e., r-row representation). The column-DFTs and the twiddle multiplications for successive decomposition stages can also be identified as radix-r butterflies with twiddles applied at the output (see Figure 1c).

2.3.2 Decimation-in-time

If, on the other hand, we let M=2 and L=N/2, we will obtain a (N/2)-row by 2-column matrix as shown in Table VIII.

Table VIII: Two-dimensional arrangement of time samples for the DIT process.

ℓ \ m	0	1
0	x_0	x_1
1	x_2	x_3
2	x_4	x_5
. . .	. . .	. . .
$\frac{N}{2} - 1$	x_{N-2}	x_{N-1}

Since the input (time) sequence is decimated into even and odd sequences, this algorithm is called the decimation-in-time process.

In general, the DIT process refers to the decimation of a sequence using an (N/r)-row by r-column representation of the original sequence. In this case, the row-DFTs are of dimension r, and the twiddle multiplications are performed before the row-DFTs. Hence, these two operations (twiddle multiplications and DFTs) can be conveniently performed by radix-r butterflies in which the twiddles are applied at the input of the r-point DFT. If N is expressible as a power of r, the column-DFTs can be further decomposed using the same generalized FFT procedures (i.e., r-column representation). The row-DFTs and the twiddle multiplications for successive decomposition stages can also be identified as radix-r butterflies with twiddles applied at the input(see Figure 1b).

2.3.3 Digit reversal

A comparison of Tables I and II shows that the generalized FFT procedure arranges time and frequency samples differently. In the case of an L-by-M representation of the N-point sequence, where the rows of the time sample matrix are formed by M contiguous time samples, the columns of the frequency sample matrix consist of L contiguous frequency samples. It follows that with time sequences fed sequentially into an FFT processor (this refers to in-place realization of the FFT in digital computer programs and hardware single pipeline FFTs) in natural order, the frequency samples do not come out in the same order. The determination of the order of the frequency samples, given the order of the time samples, is important when using the FFT. It will be shown that the frequency indices for a fixed-radix FFT algorithm can be determined from the time indices using a rule called "digit reversal". Consider a time sequence $\{x_i\}$, i=0,1,2,...,N-1. This sequence is to be processed using a fixed radix-r FFT. The index of the frequency sample occupying a position which corresponds to time sample x_i in the time sequence is determined using the following set of rules:

(i) express the index i in terms of digits of the base-r number system. The maximum number of digits required is determined by N.

(ii) Reverse the order of the digit pattern. The frequency index is given by the decimal value of the resulting number.

A simple example will suffice to clarify the procedure. Consider the 16-point time sequence $\{x_i\}$, i=0,1,2,3,...,15, to be processed by a radix-4 FFT. Suppose we want to determine the index k of a frequency sample occupying the position which corresponds to x_6 in the time sequence. First we express the number 6 in terms of digits of a base 4 number system. There are only 4 digits, 0,1,2 and 3, in a base 4 number system. The base 4 number for a decimal value of 6 is 12 (base 4). After digit reversal, the number becomes 21 (base 4). Consequently, the index for this frequency sample is 21 (base 4) = 9 (decimal).

It can easily be verified that, when the 16-point time sequence $\{x_i\}$ is processed by a radix-4 FFT, the time sequence $\{x_i\}$ is arranged in the form given by matrix A, and the resulting frequency is as given in matrix B of Eqn (14).

$$A = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} \qquad B = \begin{bmatrix} F_0 & F_4 & F_8 & F_{12} \\ F_1 & F_5 & F_9 & F_{13} \\ F_2 & F_6 & F_{10} & F_{14} \\ F_3 & F_7 & F_{11} & F_{15} \end{bmatrix}$$

In terms of base 4 digits, the indices for the above matrices become:

$$A = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{20} & x_{31} & x_{32} & x_{33} \end{bmatrix} \qquad B = \begin{bmatrix} F_{00} & F_{01} & F_{20} & F_{30} \\ F_{01} & F_{11} & F_{21} & F_{31} \\ F_{02} & F_{12} & F_{22} & F_{32} \\ F_{03} & F_{13} & F_{23} & F_{33} \end{bmatrix} \qquad (15)$$

If one compares the digit pattern of corresponding elements in the two matrices given by Eqn (15), it is seen that their order is identically reversed.

When the radix r is equal to 2, the above procedure is called "bit reversal". The bit reversal procedure can be demonstrated by applying the generalized FFT procedure successively to the N-point sequence and noting the position of the time and frequency samples when they are decomposed into two-dimensional arrays. The bit reversal is seen as a result of the transposition of the time and frequency matrices at each stage of decomposition. The details of bit reversal may be found in [4].

3. REDUCING THE COMPUTATIONAL REQUIREMENTS FOR THE DFT ALGORITHM AND ITS PROCESSING COMPONENTS

3.1 Reduction of computational effort

One might wonder, what is to be gained from the development of the elaborate procedures in Table V for computing a generalized FFT. We are now in a position to show that the computational effort is reduced substantially by employing these procedures. Let us derive the total number of complex multiplications and additions required for the generalized FFT algorithm. In step (iii) of Table V, there are M L-point DFTs. Since $(L-1)^2$ multiplications and L(L-1) complex additions are required for an L-point DFT, $M(L-1)^2$ multiplications and ML(L-1) additions are required to carry out step (iii). Step (v) consists of L M-point DFTs. Consequently, $L(M-1)^2$ multiplications and LM(M-1) additions are required by this computation. Since the twiddle factor in step (iv) is equal to unity for m=s=0, the effective number of multiplications is (L-1)(M-1). The number of operations required for these three steps is the sum of the above, yielding:

(a) Total number of complex multiplications for the generalized FFT = $M(L-1)^2 + (L-1)(M-1) + L(M-1)^2$ (16)

and

(b) Total number of complex additions for the generalized FFT = $ML(L-1) + LM(M-1)$ (17)

We can form the ratios of the multiplications and additions required for a DFT to that required for the generalized FFT algorithm. The ratio of number of complex multiplications is given by:

$$R_m = \frac{M(L-1)^2 + (L-1)(M-1) + L(M-1)^2}{(LM-1)^2} \tag{18}$$

and the ratio of number of complex additions is:

$$R_a = \frac{ML(L-1) + LM(M-1)}{LM(LM-1)} \tag{19}$$

In the limit of very large L and M, these two ratios reduce to:

$$R_m = R_a = \frac{1}{L} + \frac{1}{M} \tag{20}$$

From Eqn (20) it follows that the number of complex multiplications and additions required for the generalized FFT algorithm is only a small fraction of those required by direct computation when L and M are very large.

3.2 Radix-r FFT and radix-r butterfly

If L and M are not themselves prime numbers, they can each be decomposed into a product of two numbers, and further saving in computational effort can be realized. As a matter of fact, if N is a power of an integer r, then the DFT of an N-point sequence can be decomposed into a number of similar stages composed of radix-r butterflies. A fixed-radix FFT algorithm is one which employs butterflies of a single radix value, and a mixed-radix FFT algorithm is one which employs two or more radix values.

Using the generalized FFT procedure, we can first decompose the N-point sequence into an r-row by (N/r)-column matrix. Let us consider the number of multiplications required. Since there are r rows and (N/r) columns, we must perform (N/r) radix-r DFTs, multiply the result by the elements in an r-by-(N/r) twiddle matrix, and then perform r radix-(N/r) DFTs. If r is a prime number, it cannot be decomposed further. Consequently, the radix-r DFTs must be computed directly. Let M_r be the number of complex multiplications required to directly compute the r-point DFT. We have, for the number of multiplications required for the N-point DFT:

$$M_N = \frac{N}{r} M_r + (r-1)\left(\frac{N}{r} - 1\right) + r\, M_{(N/r)} \qquad (21)$$

$\frac{N}{r}$ (N/r) r-point DFTs; M_r; $(r-1)\left(\frac{N}{r} - 1\right)$ Twiddle Multiplications; $r\, M_{(N/r)}$ r (N/r)-point DFTs

Since N is a power of r, N/r can be written as the product r x N/r^2. This decomposition can be applied successively, starting with Eqn (21), giving the following result:

$$M_N = \frac{N}{r} M_r + (r-1)\left(\frac{N}{r} - 1\right) + rM_{\left(\frac{N}{r}\right)}$$

$$= \frac{N}{r} M_r + (r-1)\left(\frac{N}{r} - 1\right) + r\left[\frac{N}{r^2} M_r + (r-1)\left(\frac{N}{r^2} - 1\right) + rM_{\left(\frac{N}{r^2}\right)}\right] \qquad (22)$$

$$= \frac{N}{r} M_r + (r-1)\left[\frac{N}{r} - \sum_{j=0}^{p-1} r^j\right] + r^p M_{\left(\frac{N}{r^p}\right)}$$

where p signifies the pth stage decomposition

This decomposition process cannot continue indefinitely. It will stop when $N/r^p = r$. From this condition, it follows that:

$$p = \log_r N - 1 \qquad (23)$$

If one includes the initial stage of decomposition, called the 0^{th} stage, it can be concluded that, if N is a power of r, there are $\log_r N$ radix-r butterfly stages. The substitution of Eqn (23) into Eqn (22) yields:

$$M_N = \frac{N}{r} M_r \log_r N + \frac{N(r-1)}{r}(\log_r N - 1) - (r-1) \sum_{j=0}^{\log_r N - 2} r^j \qquad (24)$$

Equation (24) is valid when N is a power of the prime number r.

There are two radix values which have a special significance when computing butterflies, namely, 2 and 4. These two cases are significant because in both cases, the DFT portion of the butterfly does not require any multiplications.

3.2.1 Radix-2 butterfly and 2-point DFT

For a 2-point sequence, the maximum radix value is 2. We have from Eqn (1):

$$F_k = x_0 + x_1 \exp(-j2\pi k/2) \qquad k=0,1$$

or

$$F_0 = x_0 + x_1$$
$$F_1 = x_0 - x_1 \qquad (25)$$

Consequently, a 2-point DFT, which forms part of a radix-2 butterfly, requires only one complex addition and subtraction. The symbolic and schematic diagrams of a radix-2 butterfly are shown in Figure 2a and 2b, respectively.

We can now derive an expression which will give an accurate estimate of the number of multiplications required for computing the DFT of an N-point sequence, using radix-2 butterflies. In Eqn (21), $M_r=0$ when r=2, since a 2-point DFT does not require any multiplications. In addition, each twiddle matrix always includes one element equal to -j. Thus the count for twiddle multiplications can be reduced by one. Eqn (21) reduces to:

$$M_N = \frac{N}{2} - 2 + 2M_{(\frac{N}{2})} \qquad (26)$$

Applying the two fold decomposition successively, we have:

$$M_N = \frac{N}{2} - 2 + 2[\frac{N}{2^2} - 2 + 2M_{(\frac{N}{2^2})}]$$

$$= \frac{N}{2} - 2 + \frac{N}{2} - 2^2 + 2^2[\frac{N}{2^3} - 2 + 2M_{(\frac{N}{2^3})}] \qquad (27)$$

$$= p\frac{N}{2} - \sum_{j=1}^{p} 2^j + 2^p M_{(\frac{N}{2^p})}$$

The decomposition terminates when $p = \log_2 N - 2$, yielding:

$$M_N = \frac{N}{2}(\log_2 N-2) - \sum_{j=1}^{\log_2 N-2} 2^j$$

$$= \frac{N}{2}\log_2 N - \frac{3}{2}N + 2 \qquad (28)$$

In Eqn (28), we have made use of the fact that when $p = \log_2 N-2$, the last term in Eqn (27) becomes zero since, as will be shown next, there are no multiplications required in computing a 4-point DFT.

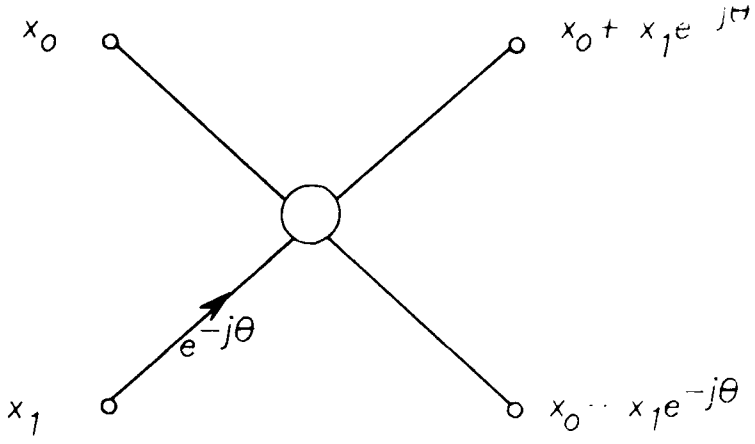


FIGURE 2. (a) RADIX-2 DIT BUTTERFLY.

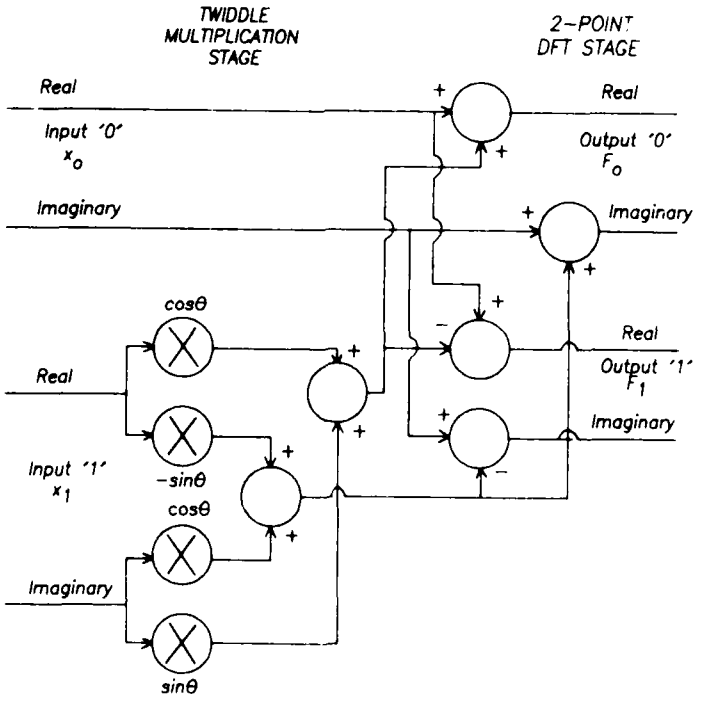


FIGURE 2. (b) SCHEMATIC DIAGRAM OF THE RADIX-2 DIT BUTTERFLY.

3.2.2 Radix-4 butterfly and 4-point DFT

A 4-point DFT does not require any multiplication. This is due to the fact that the required twiddle factors are either unity or $\pm j$. If in Eqn (21), we substitute r=4, and using the fact that one of the twiddle factor is equal to -j (thus reducing the multiplication count by one), the result is given by:

$$M_N = \frac{3}{4} N - 4 + 4M_{(N/4)} \tag{29}$$

Applying the four-fold decomposition successively, we have:

$$M_N = \frac{3}{4} N - 4 - 4[\frac{3}{4^2} N - 4 + 4M_{(N/4^2)}]$$

$$= \frac{3}{4} N - 4 + \frac{3}{4} N - 4^2 + 4^2[\frac{3}{4^3} N - 4 + 4^3 M_{(N/4^3)}] \tag{30}$$

$$= \frac{3}{4} Np - \sum_{j=1}^{p} 4^j + 4^p M_{(N/4^p)}$$

where p signifies the pth stage decomposition

The decomposition terminates when $p=\log_4 N-1$, in which case $M_{(N/4^p)} = 0$, yielding:

$$M_N = \frac{3}{4} N(\log_4 N-1) - \sum_{j=1}^{\log_4 N-1} 4^j \tag{31}$$

$$= \frac{3}{4} N\log_4 N - \frac{13N}{12} + \frac{4}{3}$$

Equations (28) and (31) do not take into account the fact that twiddle factors equal to $\pm\exp(-j\frac{3\pi}{4})$ have identical real and imaginary parts. This fact may be used to reduce the number of real multipliers in certain hardware implementations of the FFT. The symbolic diagram of a radix-4 butterfly is shown in Figure 3. A radix-4 butterfly is formed by adding a twiddle multiplication stage at the input (for DIT process) or at the output (for DIF process) of the DFT. The schematic diagram of the twiddle multiplication section of a radix-4 butterfly is similar to that of a radix2 butterfly. Each non-trivial twiddle multiplication requires four real multipliers and two real adders. The schematic diagram of the DFT portion of a radix-4 butterfly is shown in Figure 4.

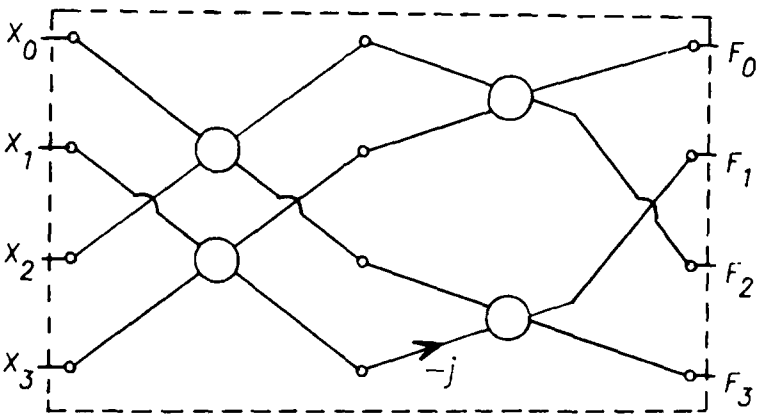


FIGURE 3. SYMBOLIC DIAGRAM OF A 4--POINT DFT.

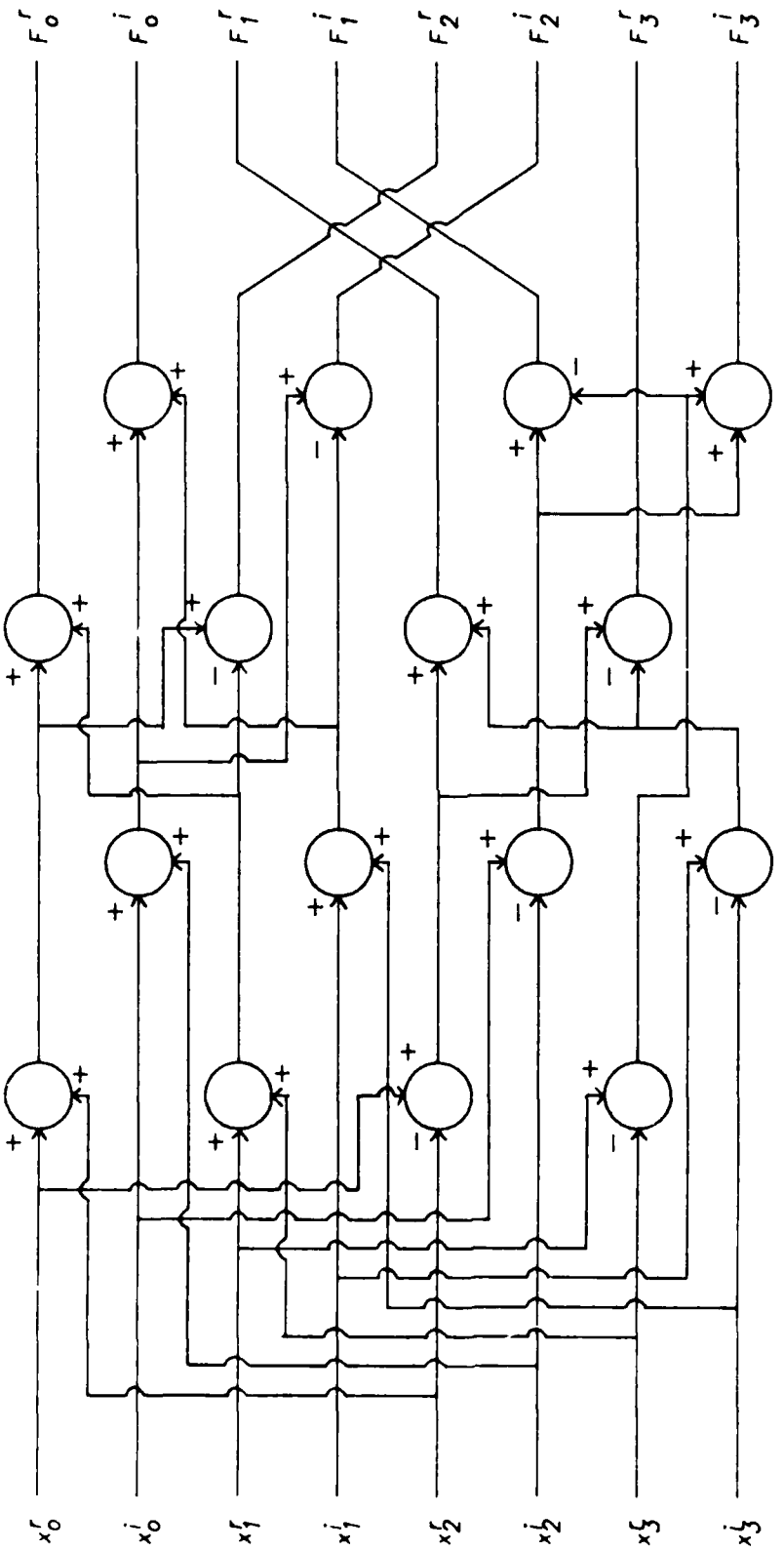


FIGURE 4. SCHEMATIC DIAGRAM OF 4-POINT DFT.

At this point it is useful to compare the number of multiplications required in computing N-point DFTs using radix-2 and radix-4 butterflies. This comparison is given in Table IX.

Table IX: Required number of complex multiplications for radix-2 and radix-4 FFTs

N	radix-2	radix-4
2	0	-
4	0	0
8	2	-
16	10	8
32	34	-
64	98	76
128	258	-
256	642	492
512	1538	-
1024	3586	2732

It seems, therefore, that whenever possible, the radix-4 configuration should be employed because the number of multiplications is less than for the radix-2 case.

3.3 Butterfly pipelining for maximum throughput rate

In most algorithms the computations take place in a sequential manner. The results are obtained by processing the input data with several cascaded stages comprising one or more signal processing components. The entire signal processing chain is called a 'pipeline', and the group of signal processing components which compute the intermediate results at each stage is called the pipeline segments. The definition of a pipeline segment is arbitrary. However, it must be carefully considered when high throughput rate is required. Up to this point, we have considered the radix-r butterfly as an integral arithmetic component(black box). This "black box" will perform one set of r complex multiplications and an r-point DFT. That is, the FFT algorithm is considered as a pipeline with radix-r butterflies as pipeline segments. In reality, however, a complex multiplication requires both real multiplications and additions. Depending on the value of the radix, a radix-r DFT may comprise several stages of complex multiplications and additions. Presumably, all the multiplications and additions can be performed by a single multiplier and a single adder. Consequently, it will take a number of processing clock cycles to obtain the butterfly result. In addition, time sharing of multipliers and adders requires temporary data storage, thereby adding overhead to the execution time. Hence for attaining the ultimate processing speed, a full complement of multipliers and adders should be used in a butterfly.

Let us consider the radix-2 butterfly whose schematic diagram is shown in Figure 2b. There are three distinct stages of arithmetic operations. The first stage is the multiplication of input '1' by the cosine and sine components of the twiddle factor, forming the direct and

cross products of the complex multiplication. The second stage is the combination of the direct and cross products to get the real and imaginary parts of the complex multiplication. The third stage is the 2-point DFT which consists of one addition and one subtraction simultaneously. Input '0' is not multiplied, hence, it must be held constant until the complex multiplication of input '1' is completed. The time interval between when data enter into and exit from a processing segment is called the propagation delay. Let T_m and T_a be the time of execution for the multiplier and adder, respectively. The minimum system processing clock interval, given by the worst case propagation delay of any section in the pipeline, must be at least $T_c = T_m + 2T_a$. This is due to the fact that these arithmetic operations take place sequentially, and new data cannot be introduced until the result is available at the output of the processing segment.

To increase the throughput rate of the radix-2 butterfly, we can insert two cascaded stages of shift registers or latches in the signal paths of input '0' and a shift register at the output of each multiplier and adder, as shown in Figure 5. At the beginning of each clock cycle, the value of input '0' is shifted into the first latch, and the output of the first latch is shifted into the second latch. Thus valid data will be available after two clock cycles for the final addition stage. Meanwhile, new data can be entered at the beginning of each new clock cycle. A butterfly with these modifications is called a pipelined butterfly. The implication of the pipeline structure is that, no matter how many steps it takes to complete an arithmetic operation, there is no need to wait until the answer is obtained before new data are entered. Consequently, the clock frequency of a properly designed pipeline signal processor is determined by the propagation delay of the slowest signal processing component. Commercially available integrated circuit (IC) multipliers and adders have attained processing speed of one 12-bit multiply (or add) per 65nsec. The worst case propagation delay is approximately equal to the multiplication (or addition) time plus the latch set up time. For a 12 bit wordlength processor, a conservative estimate of the propagation delay of a pipelined butterfly is about 100 nsec.

Similarly, the radix-4 butterfly can be pipelined to obtain a maximum throughput rate. The schematic diagram of a pipelined 4-point DFT is shown in Figure 6. There are 16 real adders and 16 latches. The effective throughput rate is one 4-point DFT per clock cycle. The component counts of the 2-point, 4-point DFTs and the trivial and nontrivial twiddle multiplications are tabulated in Table X. (pipelined structures are assumed throughout).

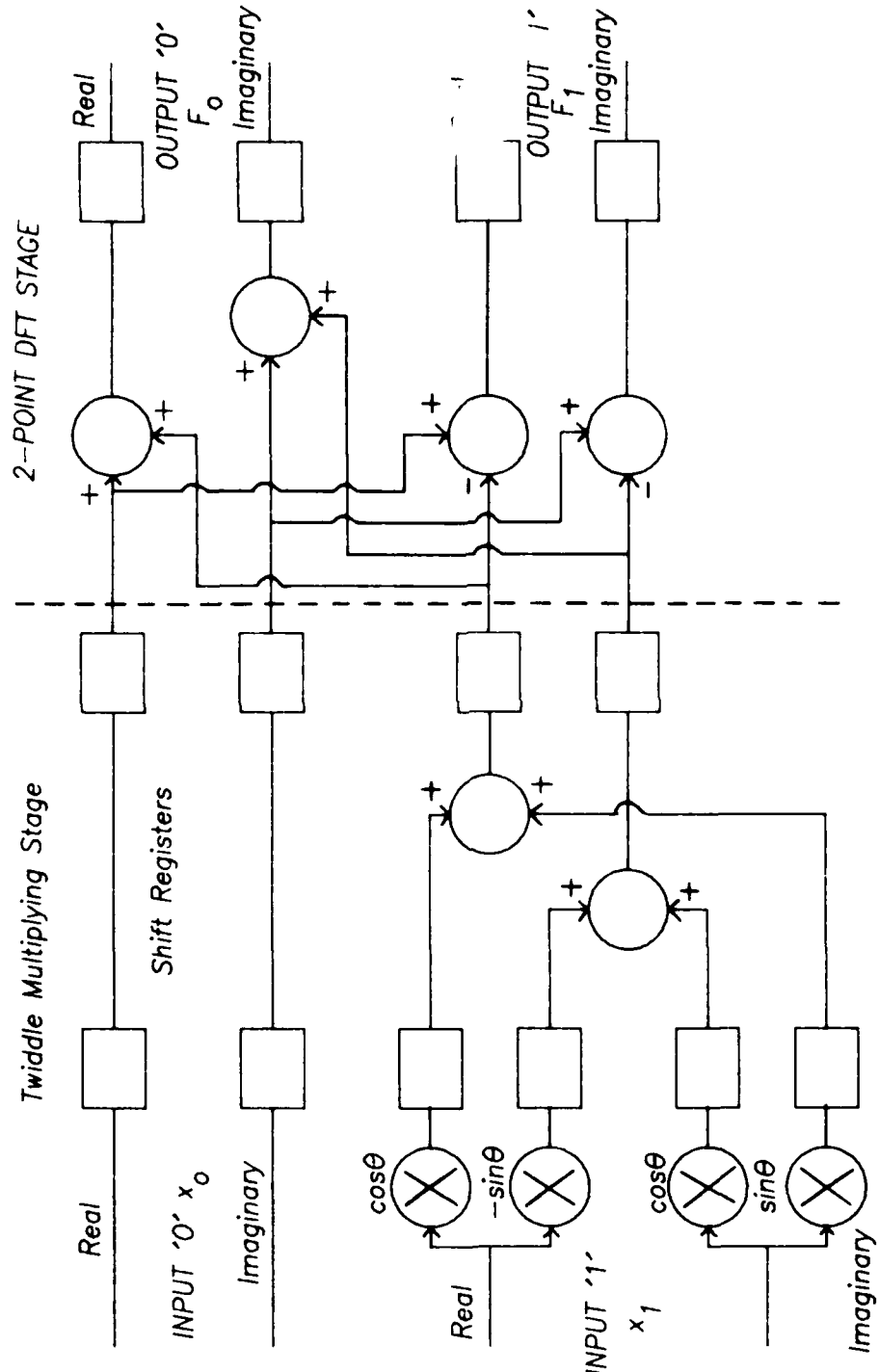


FIGURE 5. SCHEMATIC DIAGRAM OF A PIPELINED RADIX-2 BUTTERFLY.

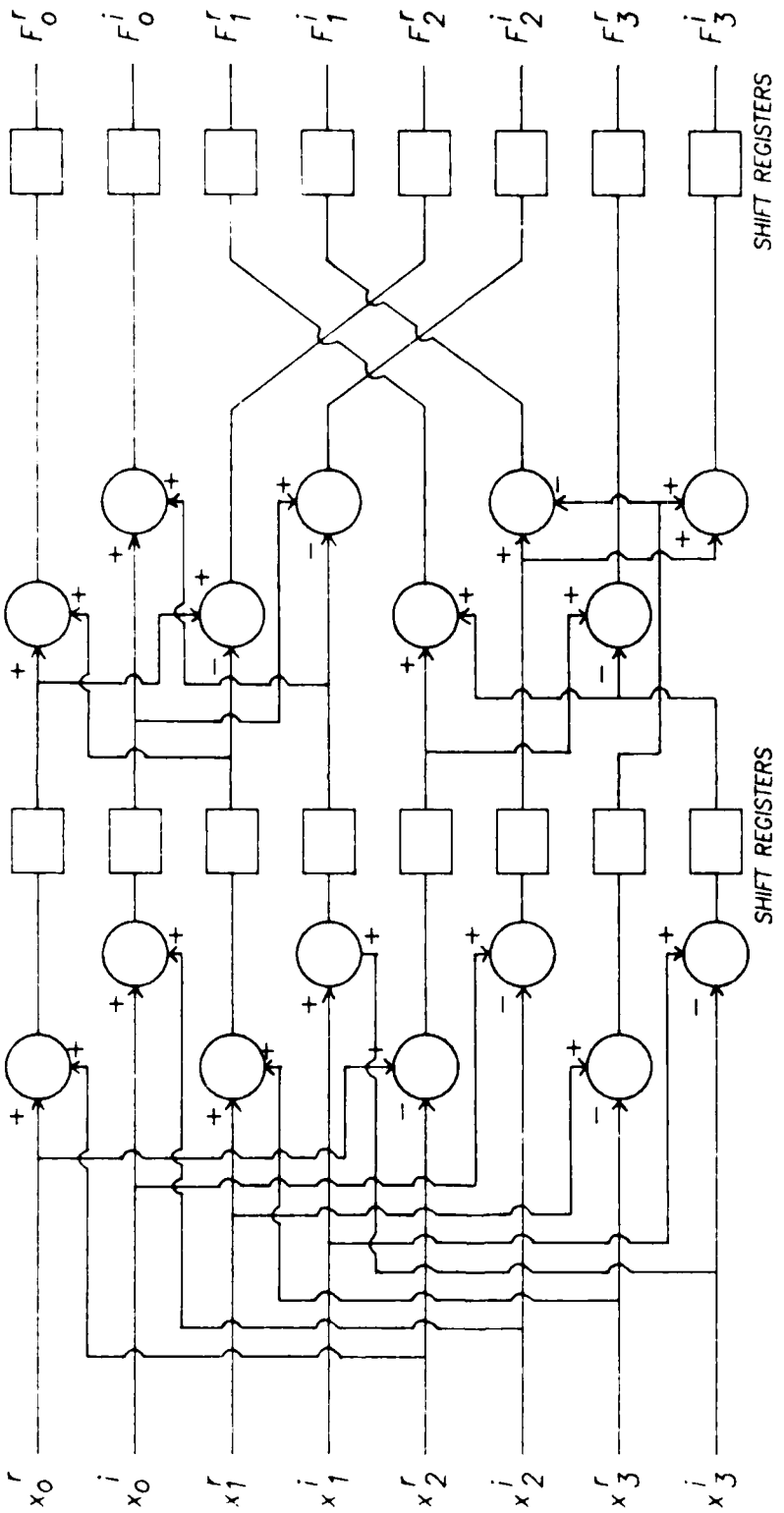


FIGURE 6. SCHEMATIC DIAGRAM OF A PIPELINED 4-POINT DFT.

Table X: Component counts of pipelined 2-point, 4-point DFTs and twiddle multiplications

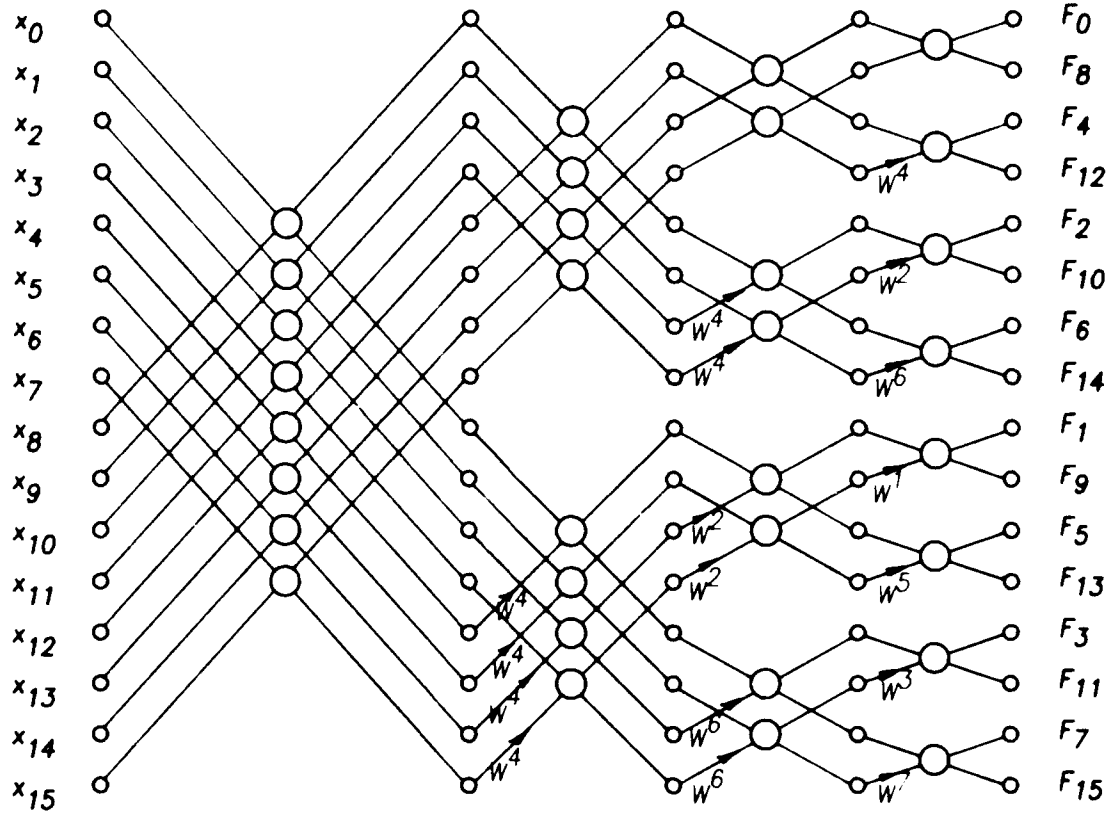
	real adders	real multipliers	latches
2-point DFT	4	-	4
4-point DFT	16	-	16
trivial twiddle multiplication	-	-	4
nontrivial twiddle multiplication	2	4	6

Table X will be referred to later to derive the approximate component counts for a number of different FFT structures.

4. Hardware implementation of the FFT algorithm

A useful tool in implementing the FFT algorithm is the signal flow diagram (see Figure 7). The signal flow diagram of a radix-r FFT algorithm can easily be obtained by following the generalized FFT procedures outlined in Table V repeatedly, so that butterflies with radix value larger than r are systematically decomposed into structures composed of radix-r butterflies only. This diagram provides a clear indication of how the DFT is broken into successive stages. For a sequential computing machine, such as a general purpose computer, the order in which the computations in each stage are carried out is relatively unimportant. However, if additional computing hardware is available, then it becomes essential to process the data in the proper sequence to ensure that the maximum throughput rate is attained.

The basic processing unit for a radix-r FFT is the radix-r butterfly. The complexity of the butterfly depends on the value of the radix. For a radix-2 butterfly, it consists of one complex multiplier and two complex adders. For a radix-4 butterfly, it consists of three complex multipliers and eight complex adders. Since each radix-r butterfly can handle r-complex samples, N/r butterfly operations must be performed in each stage, for an N-point DFT. There is a high degree of freedom in the structuring of a hardware FFT processor. The data throughput rate of an FFT depends on the amount of parallelism we incorporate into the design which is usually constrained by cost considerations. Given the signal flow diagram of an FFT algorithm in terms of radix-r butterflies, there are three basic approaches to a hardware implementation of the algorithm (see Figure 7 for an example):



W = exp(−jπ/8) Signal Paths With No Twiddle Indicator Are Assumed To Have $W^0=1$

FIGURE 7. SIGNAL FLOW DIAGRAM OF THE RADIX−2 DIT 16−POINT FFT.

(a) Employ N/r butterflies per stage so that each butterfly will only handle one set of r samples.
(b) Employ one butterfly in each stage so that each butterfly will be required to handle N/r sets of r samples per stage.
(c) Employ a single butterfly so that it must handle all

$$(\frac{N}{r} \text{Log}_r N) \text{ butterfly operations}$$

In these considerations, the hardware butterfly unit is assumed to be pipelined. We shall call implementation (a) the parallel pipeline FFT and implementation (b) the single pipeline FFT. Implementation (c) is used in most software programs and will not be considered further. The single pipeline FFT is referred to as simply pipeline FFT or cascaded FFT.

We are now in a position to examine different ways for translating the signal flow diagram of an FFT algorithm into hardware structures using commercially available IC elements. We shall use specific examples to illustrate the concepts discussed in the last two sections. The examples used are based on a 16-point FFT processor. This processor is chosen because:

(i) It has direct applications in radar signal processing
(ii) It can serve as a basic building block for implementing high speed FFT processors of large dimension
(iii) It can serve as an integral part of a high speed digital beamforming processor.

4.1 Parallel-pipeline 16-point FFT.

If the parallel pipeline structure is implemented for an N-point FFT using radix-r butterflies, there will be N/r butterflies in each stage. Assuming that hardware radix-r butterflies are available, the implementation reduces to the following two problems:

(i) Determining the inter-connections between the outputs of one stage and the inputs of the next stage.
(ii) Determining the values of the twiddle multipliers in each signal line of the butterflies.

4.1.1 Radix-2 parallel-pipeline 16-point FFT

The signal flow diagram of a radix-2 DIT 16-point FFT is shown in Figure 7. Each circle in the diagram represents a radix-2 butterfly. The symbol besides the arrow in each signal path represents the twiddle factor for that path. Signal paths with no twiddle designation are assumed to have unity twiddle.

In Figure 7, there are four stages with eight radix-2 butterflies each. Hence a parallel pipelined 16-point FFT consists of 32 hardware radix-2 butterflies. The inter-connections between outputs of the butterflies in one stage and the inputs of the butterflies in the next stage, together with the required twiddle values, are also indicated in Figure 7. We note

that the twiddle factors in the first two stage are either equal to unity or $\pm j$, therefore no real multiplications are required in these two stages. Consequently, the first two stages are composed of 2-point DFTs only. A composite component count for this structure can be obtained by counting, (i) the number of 2-point DFTs, (ii) the number of trivial and nontrivial twiddles, and (iii) by using the values given in Table X. There are 32 2-point DFTs, 10 nontrivial twiddles and 22 trivial twiddles. From Table X, we obtain an estimate of the component count which is tabulated in Table XI:

Table XI: Component count for a radix-2 16-point parallel pipeline FFT

type of component	quantity
real adders	148
real multipliers	40
latches	308(including input latches)

4.1.2 Radix-4 parallel-pipeline 16-point FFT.

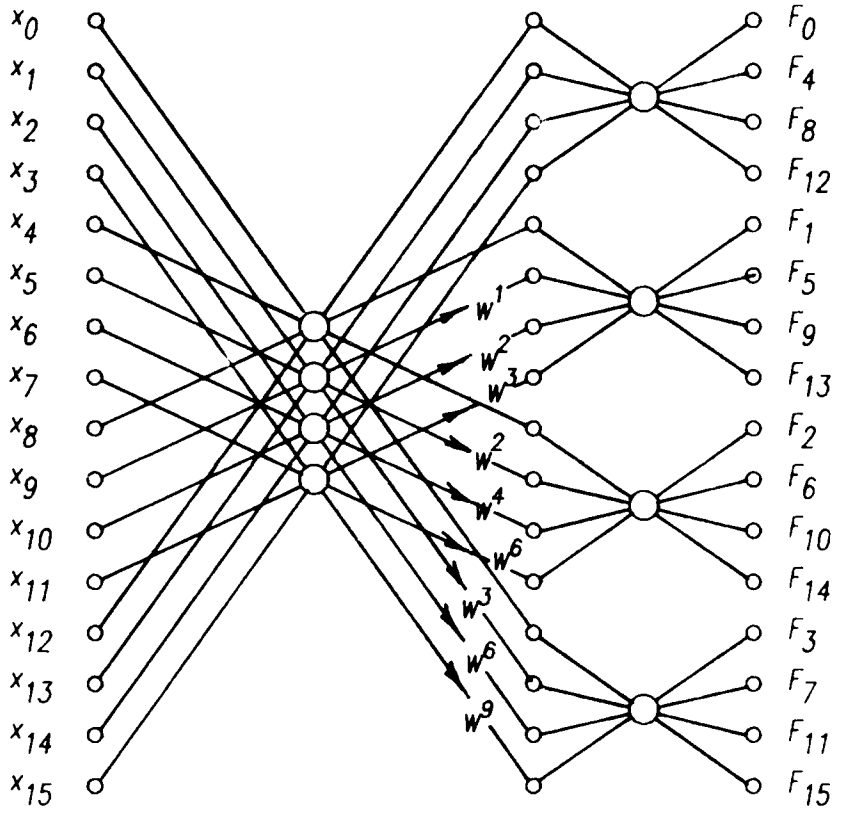
An N-point FFT can be implemented solely with radix-4 butterflies if N is a power of 4. The signal flow diagram of the radix-4 sixteen-point FFT algorithm is shown in Figure 8. There are only two stages. The samples are fed to the appropriate inputs of four separate radix-4 butterflies in the first stage, and the outputs of the first stage radix-4 butterflies are then fed to the appropriate inputs of the second stage radix-4 butterflies. The values of the twiddle multipliers in each butterfly are as indicated in Figure 8. An estimate of the composite component counts for the radix-4 parallel pipeline 16-point FFT is given in Table XII.

Table XII: Component count for a radix-4 parallel pipeline 16-point FFT

type of component	quantity
real adders	144
real multipliers	32
latches	240(including input latches)

4.2 Single-pipeline hardware FFTs

If maximum processing speed is not an absolute requirement, then the FFT algorithm can be implemented with a significant reduction in hardware. One such implementation is the so-called single pipeline FFT structure. In this structure, only one butterfly operating in time-shared mode is employed in each stage. When a signal processing component is used in a time-shared mode in a pipeline structure, the data must be introduced and processed in the proper order in order to obtain maximum efficiency. Maximum efficiency within this context refers to the state where each adder and multiplier repeats its particular function for each processing cycle. This eliminates the condition whereby a component becomes idle while awaiting the arrival of the required data.



W=exp(−jπ/8) Signal Paths With No Twiddle Indicator Are Assumed To Have $W^0=1$

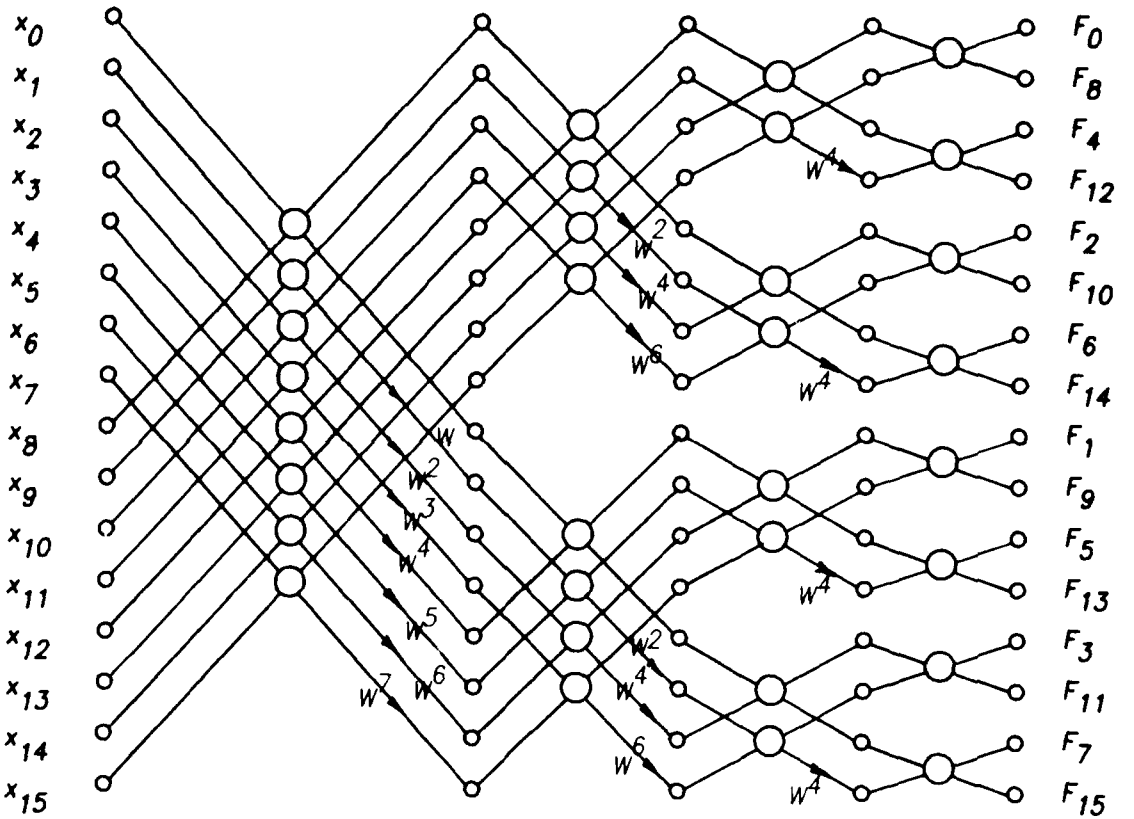
FIGURE 8. SIGNAL FLOW DIAGRAM OF THE RADIX−4 SIXTEEN POINT FFT.

4.2.1 Radix-2 DIF single-pipeline 16-point FFT

We shall use the DIF case to illustrate the radix-2 single pipeline 16-point FFT structure. The signal flow diagram for this particular FFT algorithm is shown in Figure 9. Input sample pairs $(x_0,x_8),(x_1,x_9),\ldots,(x_7,x_{15})$ are processed in the first stage. Since there is only one radix-2 butterfly in each stage, a decision has to be made with regards to which data pair to process first. Let us assume that we process the data pairs in normal order, i.e., $(x_0,x_8),(x_1,x_9),..,$etc. One simple way of feeding data to the first stage butterfly is shown in Figure 10a. There are two shift register arrays, SR0 and SR1, with eight shift registers each. The output of shift register array SR0 is connected to input '0' of the first stage radix-2 butterfly. The output of SR1 is concurrently connected to the input of SR0 and input '1' of the butterfly. One sample of the input sequence is shifted into SR1 in each clock cycle. By the end of the sixteenth clock cycle, the data pair will be aligned as shown in Figure 10b. It can be seen that this is exactly the ordering required for processing of the data.

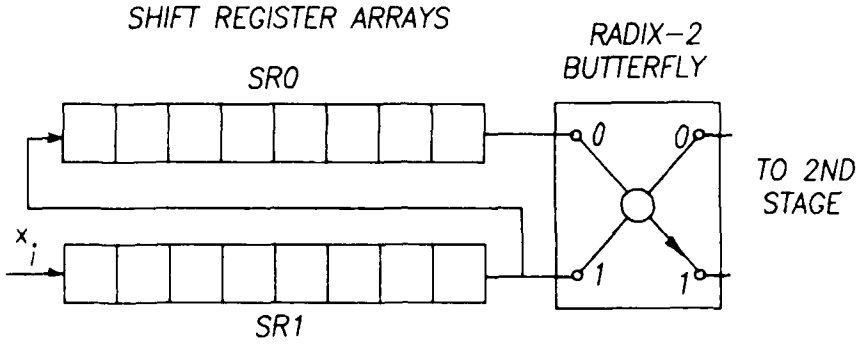
Now suppose that successive groups of sixteen samples follow immediately, then by the end of the twenty-fourth clock cycle, the data alignment will be as shown in Figure 10c. This configuration of the data corresponds to the condition where one datum of each sample pair is from the first batch, and the other datum is from the second batch. It is sometimes desirable to utilize such a condition. One example is in the filtering of a long data sequence where the data are segmented into a number of sub-sequences of fixed length N. The FFT of overlapped sequences can be used to eliminate aliasing effects[6] by discarding a number of filtered samples. On the other hand, overlapping condition is often undesirable, if the FFT is used to process distinct sequences. In this case, the FFT of two overlapped sequences constitutes invalid data. The efficiency of the processor for this configuration is effectively 50 percent.

An efficiency of 100 percent can be achieved if two separate sequences are available simultaneously. The schematic diagram of a data buffer switching network designed to enable the radix-2 butterfly to operate at 100% efficiency is shown in Figure 11. This network is sometimes called a radix-2 commutator. Its function is to direct the two separate sequences into the proper data buffer so that both sequences can be processed by the radix-2 butterfly within the time period equal to the length of the sequences. This network consists of two shift register arrays (SR0 and SR1), and two 2-to-1 multiplexer (MUX SW0 and SW1). The output of SR0 is connected to input '0' of the radix-2 butterfly. The input of SR0 is connected to the output of SW0. Consider two distinct 16-point sequences, $\{x_i\}$ and $\{y_i\}$. One sample from each sequence is read from memory simultaneously in each clock cycle. The sample from sequence $\{x_i\}$ is fed to input '0' of both SW0 and SW1. The sample from sequence $\{y_i\}$ is fed to the input of shift register array SR1. The output of SR1 is connected to input '1' of both SW0 and SW1. Switching of both SW0 and SW1 takes place every N/2(=8 for N=16) clock cycles. However, the two multiplexers do not select the same input at any given time. That is, if SW0 selects input '0', then SW1 selects input '1', and vice versa.

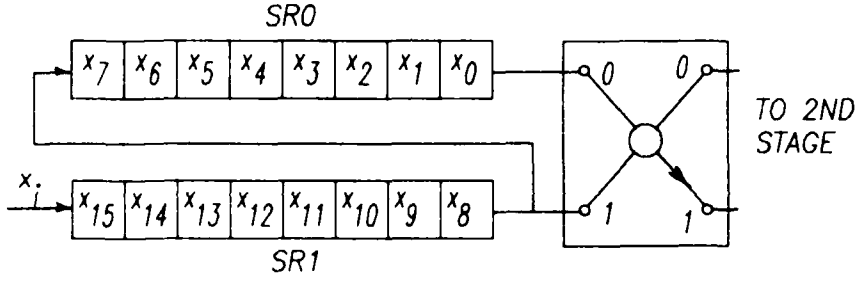


W = exp(−jπ/8) Signal Paths With No Twiddle Indicator Are Assumed To Have $W^0=1$

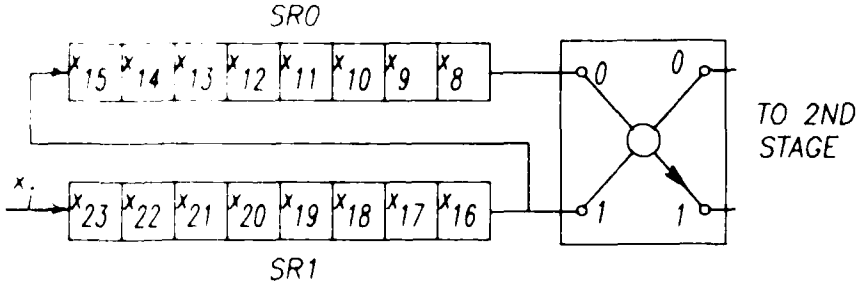
FIGURE 9. SIGNAL FLOW DIAGRAM OF THE RADIX−2 DIF 16−POINT FFT.



(a) INPUT NETWORK FOR A RADIX–2 PIPELINE 16–POINT FFT WITH SERIAL INPUT.

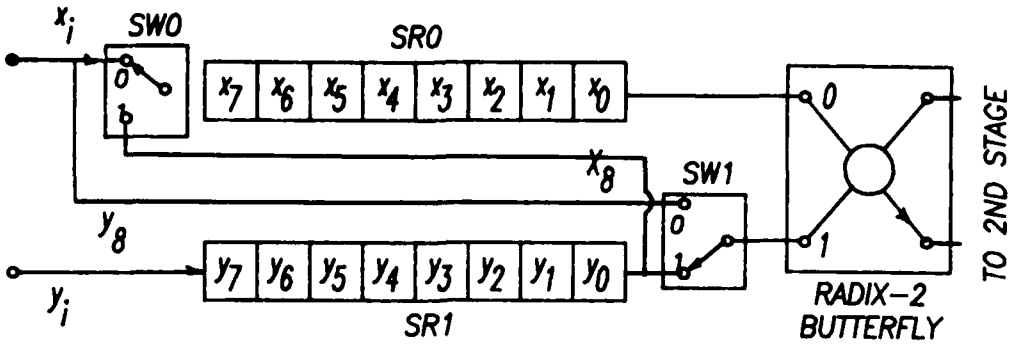


(b) DATA ARRANGEMENT AT THE BEGINNING OF CLOCK CYCLE #16.

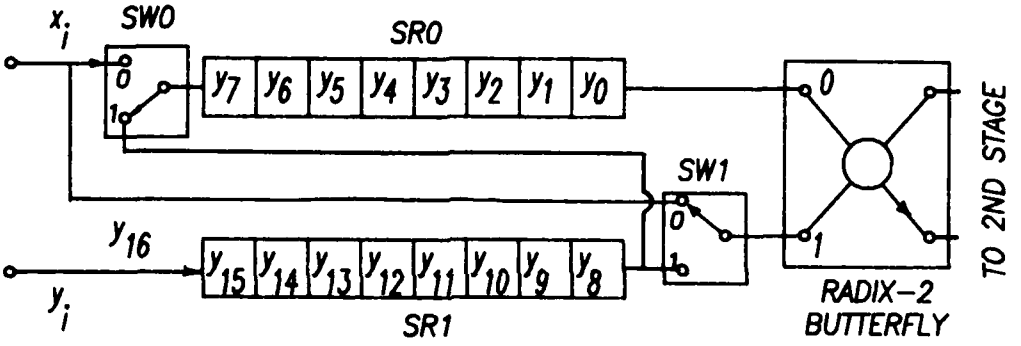


(c) DATA ARRANGEMENT AT THE BEGINNING OF CLOCK CYCLE #24.

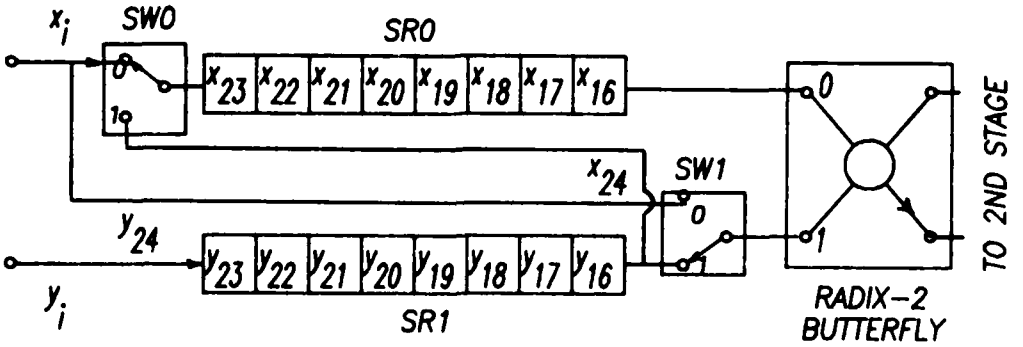
FIGURE 10.



(a) SWITCH POSITIONS REMAIN FROM CLOCK CYCLE #0 TO #7; DATA ARRANGEMENT AT THE BEGINNING OF CLOCK CYCLE #8.



(b) SWITCH POSITIONS REMAIN FROM CLOCK CYCLE #8 TO #15; DATA ARRANGEMENT AT THE BEGINNING OF CLOCK CYCLE #16.



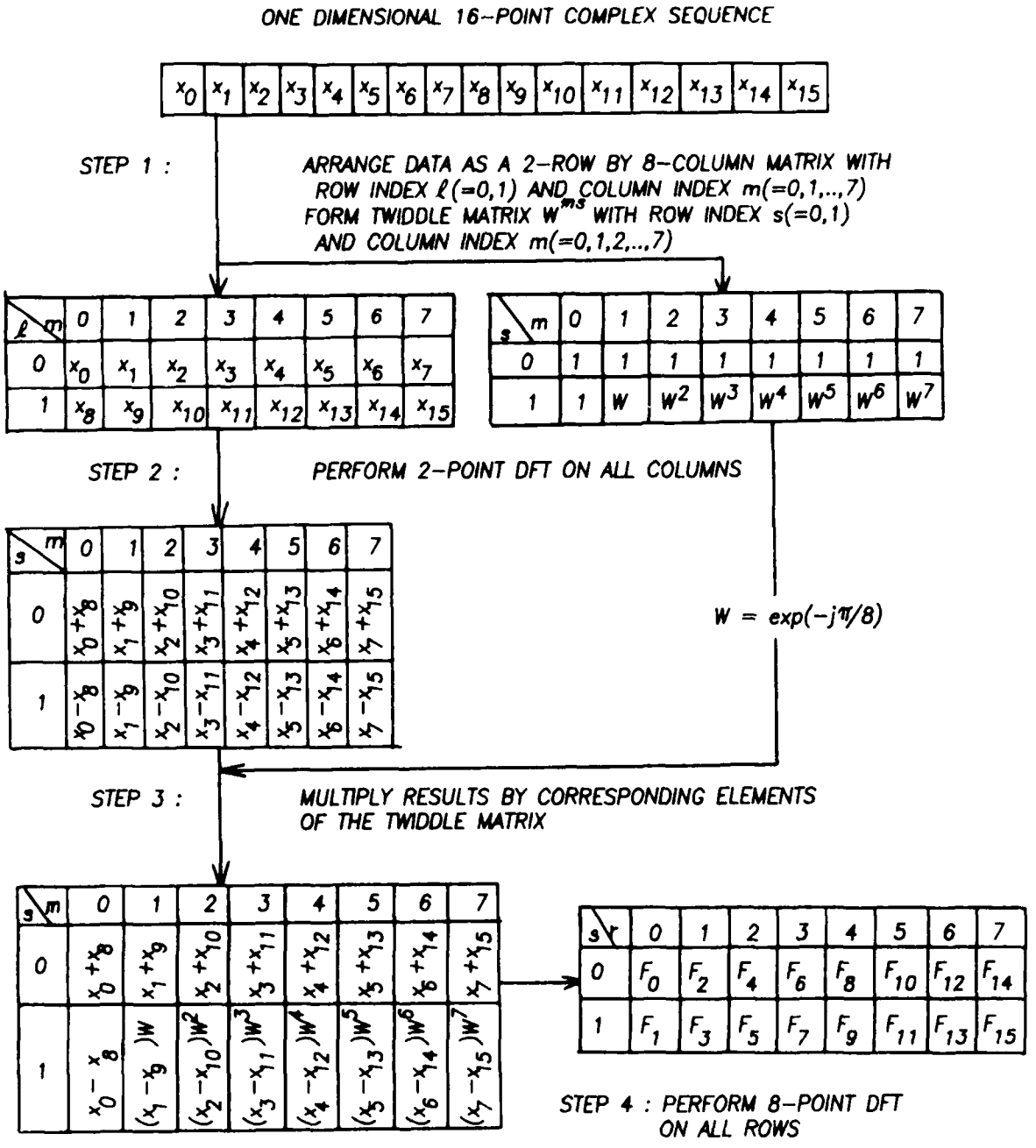
(c) SWITCH POSITIONS REMAIN FROM CLOCK CYCLE #16 TO # 23; DATA ARRANGEMENT AT THE BEGINNING OF CLOCK CYCLE #24.

FIGURE 11.

It will be shown that the overlapping of data is eliminated by the arrangement in Figure 11. At the beginning of clock cycle 0, switching takes place in both SW0 and SW1. Multiplexer SW0 select input '0' and SW1 selects input '1'. These switch connections remain in place for 8 clock cycles (i.e., until the end of clock cycle 7). Since input '0' of SW0 is connected to the signal line containing samples from sequence $\{x_i\}$, this switch connection effectively feeds the samples from sequence $\{x_i\}$ and $\{y_i\}$ into shift register arrays SR0 and SR1, respectively. The resulting data arrangement, at the end of clock cycle '7', is as shown in Figure 11b. At this time the two multiplexers are switched, with SW0 selecting input '1' and SW1 selecting input '0'. This effectively connects the input of SR0 to the output of SR1. Also input '1' of the radix-2 butterfly is now connected directly to the signal line containing samples from sequence $\{x_i\}$. If we examine the samples appearing at the inputs of the radix-2 butterfly, we find that, at the beginning of clock cycle '8', they are (x_0,x_8), followed by (x_1,x_9) in the next clock cycle, etc. This is exactly the order of data pairs called for by the signal flow diagram in Figure 9. Consequently, the radix-2 butterfly can commence processing of these data pairs starting at the beginning of clock cycle '8'.

It will take 8 clock cycles to process all 8 pairs of samples representing sequence $\{x_i\}$. Meanwhile, the contents of SR1(representing the first 8 samples of sequence $\{y_i\}$) are shifted into SR0, one sample per clock cycle. At the end of clock cycle '15', the proper sample pairs for sequence $\{y_i\}$ will be contained in the two shift register array. At this time, both multiplexers are switched once again, effectively connecting inputs '0' and '1' of the radix-2 butterfly to SR0 and SR1, respectively. The radix-2 butterfly will process the sample pairs from sequence $\{y_i\}$ beginning at clock cycle '16'. At the same time the switch connections allow samples from sequences $\{x_i\}$ and $\{y_i\}$ to be shifted into SR0 and SR1, respectively. Again it will take 8 clock cycles to process all 8 pairs of samples from sequence $\{y_i\}$. The function of the data buffer switching network is to take two simultaneous and distinct 16-point sequences $\{x_i\}$ and $\{y_i\}$ and produce two contiguous sequences of sample pairs. It can be seen that, with this data buffer switching network, the radix-2 butterfly in the first stage of a single pipeline FFT, can effectively process two distinct 16-point sequences in 16 clock cycles, after an initial delay of 8 clock cycles. If one has only a single data sequence, then it must first be divided into two sub-sequences (one comprised of data from the first half, and the other comprised of data from the second half of the original sequence). Both sub-sequences are then fed concurrently into the first stage butterfly. In this case, the radix-2 commutator in the first stage may be eliminated.

In order to gain a better understanding of the structure and operation of a single pipeline FFT processor, it is best to start with the generalized FFT procedures. In Figure 12 are summarized the steps of the generalized FFT procedures as applied to the radix-2 DIF 16-point FFT. In steps No.1 and 2, the samples of sequence $\{x_i\}$ are arranged into a 2-row by 8-column matrix. Two-point DFTs are performed on all 8 columns, and the results multiplied by the corresponding elements in the twiddle matrix. In a single pipeline FFT, these steps are performed by the first stage



Note : If Time Sequence Is Decomposed Into Rows Of Contiguous Samples, Then Then The Frequency Data Will Be Decomposed Into Columns Of Contiguous Samples.

FIGURE 12. GENERALIZED FFT PROCEDURES APPLIED TO THE RADIX-2 DIF 16-POINT FFT CASE.

butterfly. There is no problem in obtaining the proper data pair arrangement at the first stage, since the data are usually fetched from random access memory. The radix-2 butterfly will perform the 2-point DFTs in natural order, i.e., (x_0,x_8), (x_1,x_9),..,etc. The proper twiddle factor for each column DFT is read from random access memory to produce the desired result. In Step No.3, 8-point DFTs of the results of Step No.1 and 2 stored in the two rows of the data matrix are to be performed. Notice that the samples of the top and bottom rows come out of outputs '0' and '1' (see Figure 11) of the first stage butterfly, respectively. Thus the radix-2 butterfly in the first stage is effectively supplying the radix-2 butterfly in the second stage with two distinct 8-point sequences concurrently. In the previous section, it has been shown that a radix-2 butterfly is capable of processing two concurrent 16-point sequences in 16 clock cycles, when used in conjunction with a radix-2 commutator. Consequently, we may consider the radix-2 butterfly in the second stage of a 16-point, single-pipeline FFT as that of the first stage of an 8-point single-pipeline FFT.

This argument may be generalized to obtain the design of an N-point single pipeline FFT. An N-point single-pipeline radix-2 FFT is composed of $\log_2 N$ radix-2 butterflies. In order to operate at 100% efficiency, the N-point sequence must be divided into two (N/2)-point concurrent sequences corresponding to the first and second halves of the original N-point sequences, respectively. A data buffer switching network, called the radix-2 commutator, is inserted between the butterflies. This device effectively converts two concurrent N-point sequences into two contiguous (N/2)-data pair sequences. The length of the shift register arrays and the switching period of the commutator is equal to half of the length of the sequence at the input of the commutator. For example, at the output of the first radix-2 butterfly of a 16-point, single-pipeline, radix-2 FFT, the length of the data sequence is 8. Consequently, the switching period and the length of the shift register arrays of the commutator in the second stage are 4. Since the radix-2 commutator effectively converts two N-point concurrent sequences into two contiguous data pair sequences of length N/2, the switching period and the length of the shift register arrays of the commutator in each stage is half that of the preceding stage. The twiddle multipliers in each stage will vary according to the signal flow diagram; however, they will follow a definite pattern. Hence, they can be stored in a shift register array and entered into the multiplier in each clock cycle. The complete schematic diagram of the radix-2 DIF, single pipeline, 16-point FFT processor is shown in Figure 13. Except for the order of the twiddle multiplications and the switching frequency of the MUX, the structure of the single-pipeline, radix-2 DIT and DIF FFT processors is identical.

We can obtain an estimate of the hardware requirement for a single-pipeline, radix-2 16-point FFT processor. Since there are $\log_2(16)=4$ stages, only four radix-2 butterflies are required. In the last stage, all twiddle multipliers are unity, therefore, no multipliers are required. In the third stage, although all twiddle values are trivial (1 and $\pm j$), there is a requirement for the selection of one of the three values. Consequently, it is simpler to employ a multiplier. Consequently, three radix-2 butterflies and one 2-point DFT are required. We shall assume that the shift register arrays are implemented with individual latches. The number

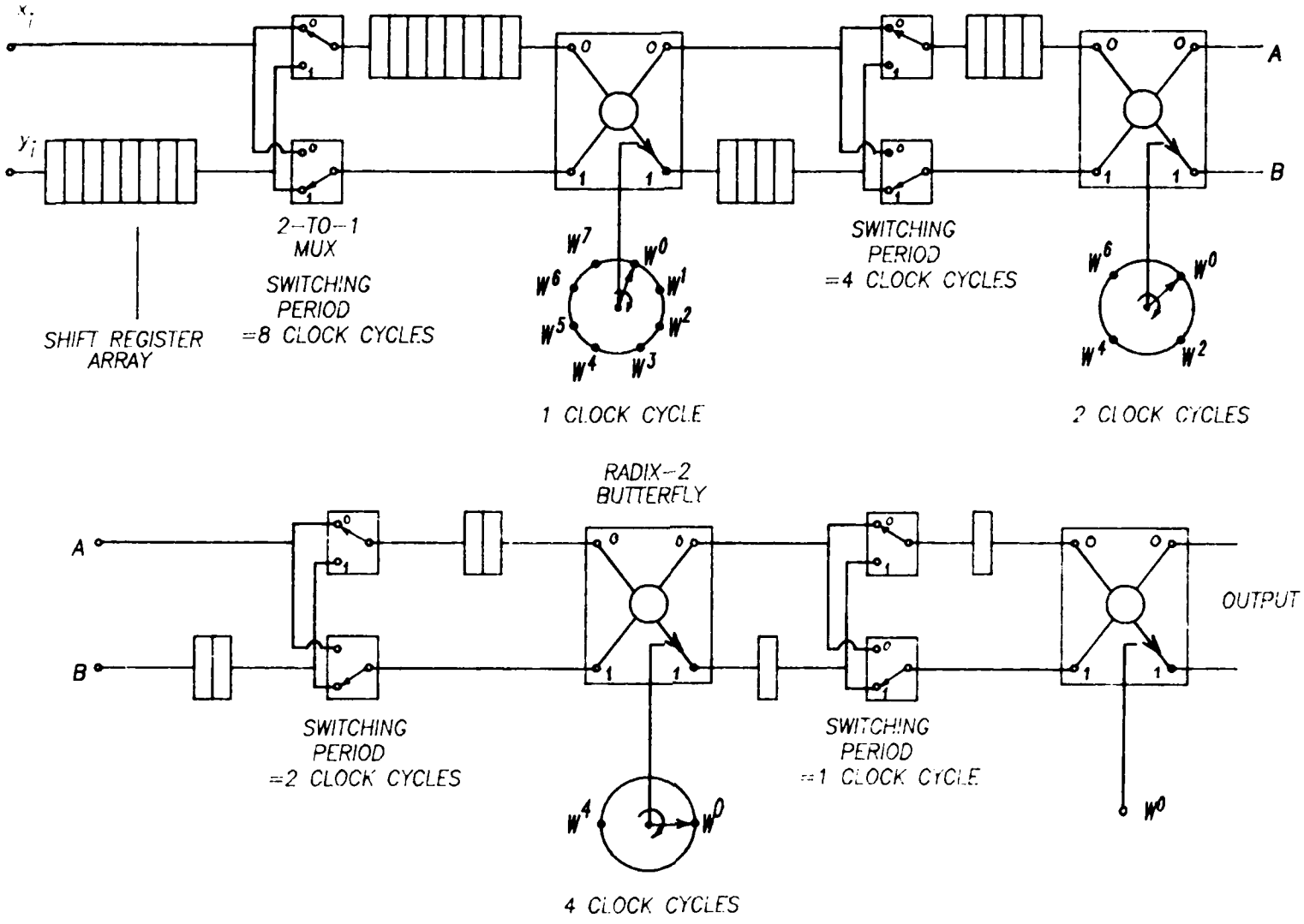


FIGURE 13. SCHEMATIC DIAGRAM OF A SINGLE PIPELINE RADIX–2 DIF 16–POINT FFT.

of latches in the commutator can be obtained from an inspection of Figure 13. In most instances, input data are read from random access memory. Consequently, no commutator is required at the input stage. However, input latches are required at the input to hold the value constant over one clock cycle. An estimate of the components required for a single-pipeline, radix-2 16-point FFT are given in Table XIII.

Table XIII: Composite component counts of the single pipeline, radix-2 16-point FFT

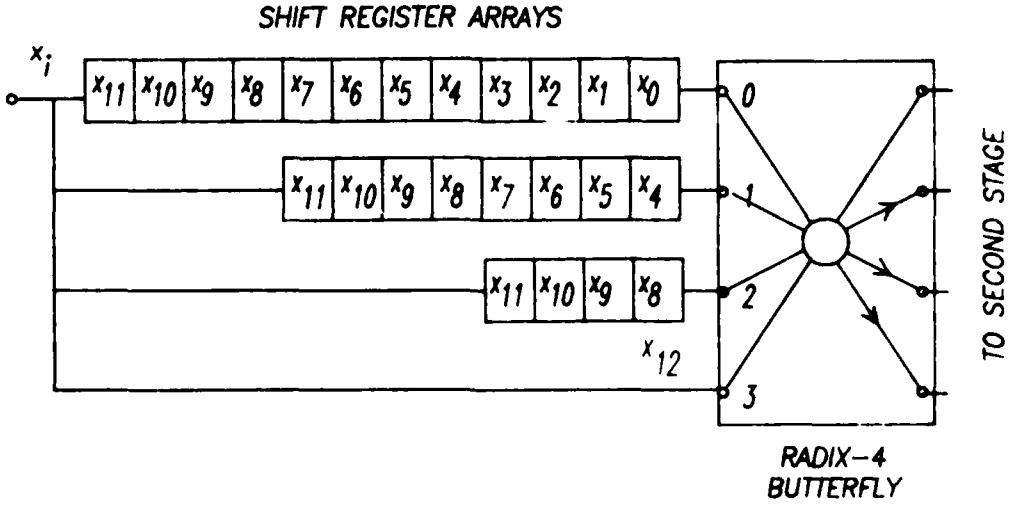
Type of component	quantity
real adders	22
real multipliers	12
latches	78
2-to-1 Multiplexers	12

The values given in Table XIII take into account the fact that each sample has a real and an imaginary part. The memory and multiplexers required for storing and switching the variable twiddle factors are not included.

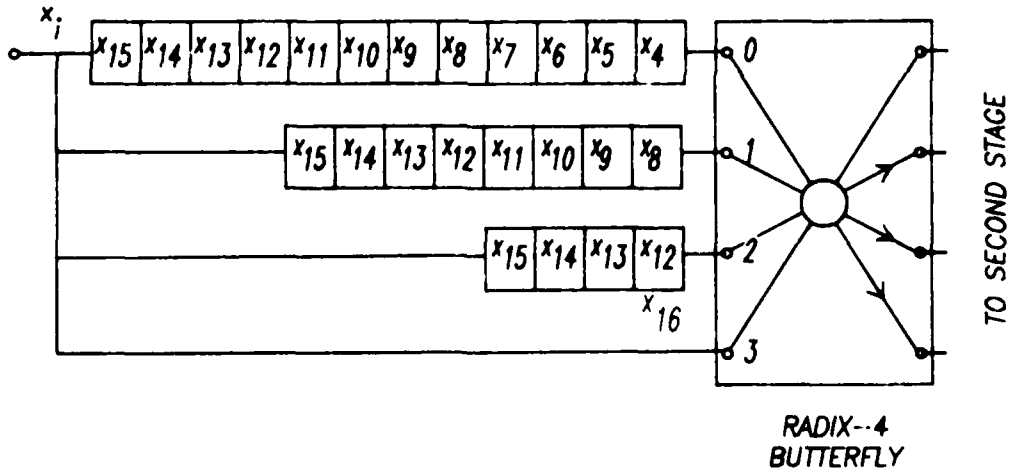
4.2.2 Radix-4 single-pipeline 16-point FFT

The 16-point FFT can also be implemented as a single-pipeline radix-4 structure, since the number 16 can be expressed as a power of 4. As indicated in Figure 8 when a DFT is performed on $\{x_i\}$, the set of samples, (x_0,x_4,x_8,x_{12}), must appear concurrently at the input of the radix-4 butterfly of the first stage, followed by the set (x_1,x_5,x_9,x_{13}), etc. For a single 16-point sequence, the above data arrangement can be obtained by inserting a data buffer between the input and the radix-4 butterfly, as shown in Figure 14. The input is connected to input terminals '0', '1', '2' and '3' of the radix-4 butterfly via shift register arrays of lengths 12, 8, 4 and 0, respectively. Since samples x_0, x_4 and x_8 are separated from sample x_{12} by 12, 8, and 4 clock cycles respectively, These shift register arrays provide the proper time delays for each of the input terminals of the radix-4 butterfly. Assuming that sequence $\{x_i\}$ is introduced to the network starting at clock cycle '0', the data arrangement at the end of clock cycle '11' is shown in Figure 14a. It can be seen that the data are aligned as required in Figure 8. Consequently, At the beginning of clock cycle '12', the radix-4 butterfly can start processing the samples appearing at its inputs. After four clock cycles, all samples from sequence $\{x_i\}$ will be processed. The data arrangement at this time, i.e., at the end of clock cycle '16', is shown in Figure 14b. Notice that three of the four samples appearing at the inputs of the butterfly have been processed before. This constitutes a 75% overlap in the data between two contiguous 16-point sequences. This overlapping is undesirable since the effective efficiency of the processor is only 25%

It can be shown that 100% efficiency can be realized if four distinct sequences are available concurrently, and if proper input buffering is employed. Consider four distinct 16-point sequences, $\{x_i\}$, $\{y_i\}$, $\{z_i\}$



(a) DATA ARRANGEMENT OF THE RADIX–4 INPUT NETWORK AT THE END OF CLOCK CYCLE No. 11.



(b) DATA ARRANGEMENT OF THE RADIX–4 INPUT NETWORK AT THE END OF CLOCK CYCLE No. 15.

FIGURE 14.

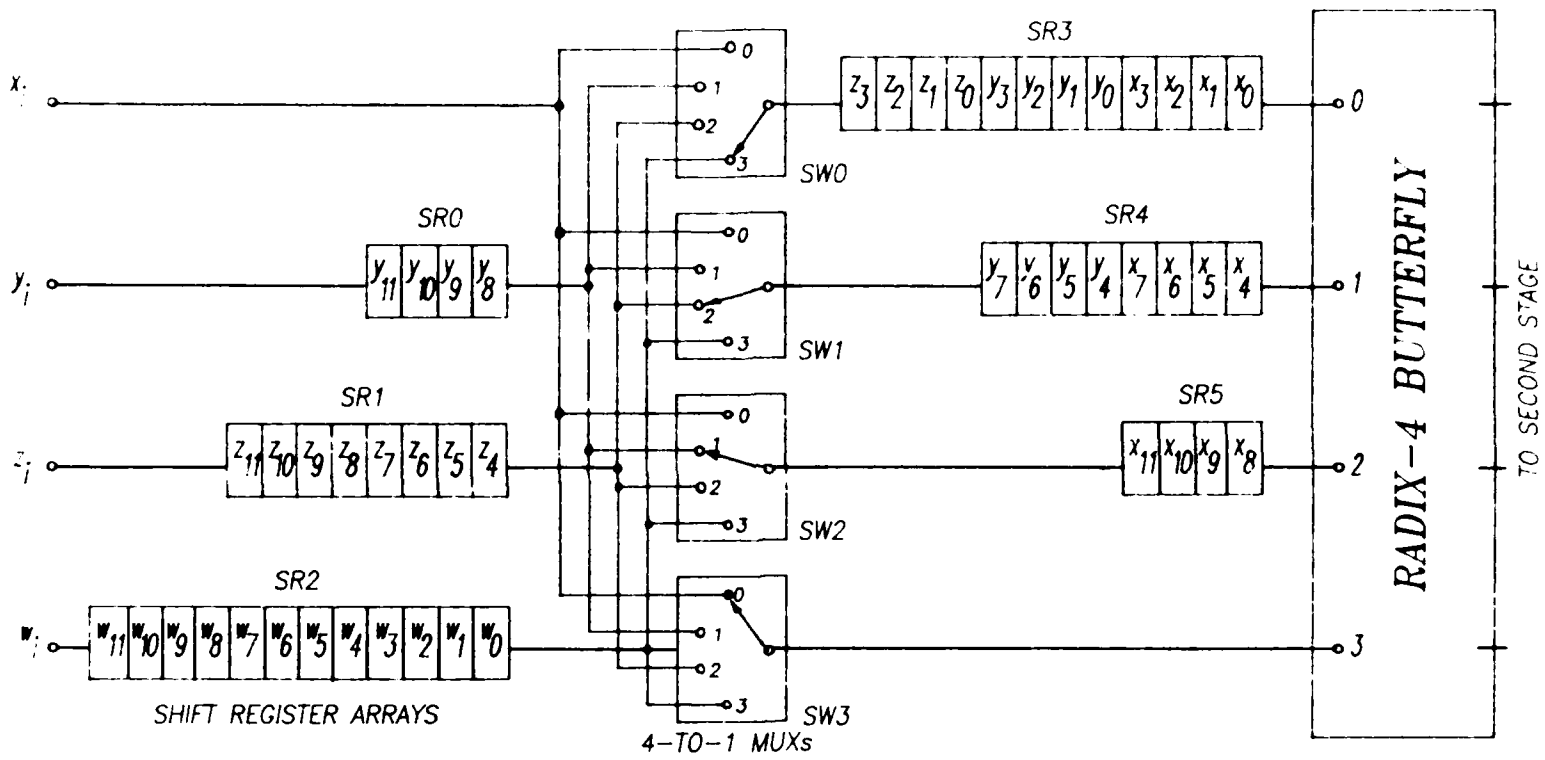


FIGURE 15. DATA SWITCHING NETWORK FOR THE FIRST STAGE OF A SINGLE PIPELINE RADIX-4 16-POINT FFT.

and $\{w_i\}$, i=0,1,2,..,15. The schematic diagram of a data buffer switching network is shown in Figure 15. This network consists of four 4-to-1 multiplexers and three pairs of shift register arrays of lengths 12, 8 and 4, respectively. We shall call this network a radix-4 commutator, because it enables a radix-4 butterfly to process four separate 16-point sequences in 16 clock cycles.

There are four inputs to the commutator which in Figure 15 are numbered 0, 1, 2 and 3 from top to bottom. The input data distribution is the following: (i) data from sequence $\{x_i\}$ are fed to input '0' of all four MUXs directly, (ii) Data from sequence $\{y_i\}$ are fed to a four-stage shift register array (SR0) with the array's output connected to input '1' of all four MUXs, (iii) Data sequence $\{z_i\}$ is fed to an eight-stage shift register array (SR1) whose output is connected to input '2' of all four MUXs, and (iv) Data sequence {wi} is fed to a twelve-stage shift register array (SR2) with the array's output connected to input '3' of all four MUXs.

The output data distribution is the following: (i) the output of MUX SW0 is connected to a 12-stage shift register array(SR3) whose output is connected to input '0' of the radix-4 butterfly, (ii) the output of SW1 is connected to an 8-stage shift register array (SR4) with the array's output connected to input '1' of the butterfly, (iii) the output of SW2 is connected to a 4-stage shift register array (SR5) with the array's output connected to input '2' of the butterfly, and (iv) the output of SW3 is connected directly to input '3' of the butterfly.

A description is now given of the operation of a radix-4 commutator during the first stage of a single-pipeline radix-4 FFT. Let us assume that data are introduced to the commutator from all four data sequences beginning at clock cycle No.0. The MUXs in the commutator operates with a switching period equal to 1/4 of the length of the data sequence. In this case, N=16, therefore, the MUXs will switch once every four clock cycles. The switch connections for the four MUXs are tabulated in Table XIV over a period of 20 clock cycles.

Table XIV: Switching connections for the MUXs as a function of clock cycles.

MUX	clock cycle				
	0,1,2,3	4,5,6,7	8,9,10,11	12,13,14,15	16,17,18,19
SW0	0	1	2	3	0
SW1	3	0	1	2	3
SW2	2	3	0	1	2
SW3	1	2	3	0	1

At clock cycle '0', all the shift register array are cleared. MUXs SW0, SW1, SW2 and SW3 select inputs '0', '3', '2' and '1', respectively. These connections remain unchanged until the end of clock cycle '3'. During

the first four clock cycles, the first 4 samples of sequences $\{x_i\}$, $\{y_i\}$, $\{z_i\}$, and $\{w_i\}$ are loaded into shift register arrays SR3, SR0, SR1 and SR2, respectively. At the beginning of clock cycle '4', MUXs SW0, SW1, SW2 and SW3 are switched to inputs '1', '0', '3', and '2', respectively. This results in the following connections: (i) the input of SR3 is connected to the output of SR2, (ii) the input of SR4 is connected to the output of SR1, (iii) the input of SR5 is connected to the output of SR0, and (iv) input '3' of the butterfly is connected directly to data line $\{x_i\}$. Following the switching sequence given in Table XIV, one can easily verify that the data arrangement at the end of clock cycle '11' is as shown in Figure 14. It can be seen that the samples are now permuted in the manner as required by the signal flow diagram (Figure 8). Therefore, the radix-4 butterfly will start by processing the data from sequence $\{x_i\}$. All the data for sequence $\{x_i\}$ will be processed after four clock cycles. After four clock cycles, the samples of sequence $\{y_i\}$ will appear at the input of the butterfly with the required permutation. Thus the function of a radix-4 commutator is to convert 4 concurrent N-point sequences into four contiguous sequences of length N/4. Each entry in the resulting sequence consist of four samples. This conversion is illustrated in Figure 16.

The structure and operation of a single pipeline radix-4 FFT can now be described. In Figure 17, the steps of the generalized FFT procedure, as applied to the radix-4 16-point FFT, are summarized. Steps 1 and 2 are performed by the first stage radix-4 butterfly. Since the data are processed in natural order (i.e., column (x_0,x_4,x_8,x_{12}) is processed first, followed by (x_1,x_5,x_9,x_{13}), etc.), each output of the radix-4 butterfly will produce the required row-data in step 3. These row-data are distinct sequences of length N'=N/4. In the previous section, it was shown that, by employing a radix-4 commutator, a radix-4 butterfly is capable of processing four concurrent N-point sequences in N clock cycles. It follows that the processing outlined in Figure 17 can be performed using similar sections of radix-4 butterflies and commutators. The length of the shift register arrays and the switching period of the commutator are one quarter as long as the corresponding element in the preceding section. The complete schematic diagram of a single pipeline radix-4 16-point FFT is shown in Figure 18.

Let us now derive an estimate of the hardware requirements for the processor given in Figure 18. There are two radix-4 DFTs with 16 real adders and 16 latches for pipeline operation. The twiddle multipliers may be attached to the first- or second-stage radix-4 DFT to form a radix-4 butterfly. In any case, there will be one trivial twiddle and three non-trivial twiddles. The number of multiplexers and latches in the commutator may be obtained by inspection of Figure 18. Assuming that input data are read from random access memory in the proper order, the commutator at the input stage can be eliminated. The component count for a radix-4 single pipeline 16-point FFT is given in Table XV:

INPUT : FOUR CONCURRENT SEQUENCES

x_{15} x_{14} x_{13} x_{12} x_{11} x_{10} x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0

y_{15} y_{14} y_{13} y_{12} y_{11} y_{10} y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0

z_{15} z_{14} z_{13} z_{12} z_{11} z_{10} z_9 z_8 z_7 z_6 z_5 z_4 z_3 z_2 z_1 z_0

w_{15} w_{14} w_{13} w_{12} w_{11} w_{10} w_9 w_8 w_7 w_6 w_5 w_4 w_3 w_2 w_1 w_0

OUTPUT : FOUR CONTIGUOUS SETS OF DECIMATED SEQUENCES.

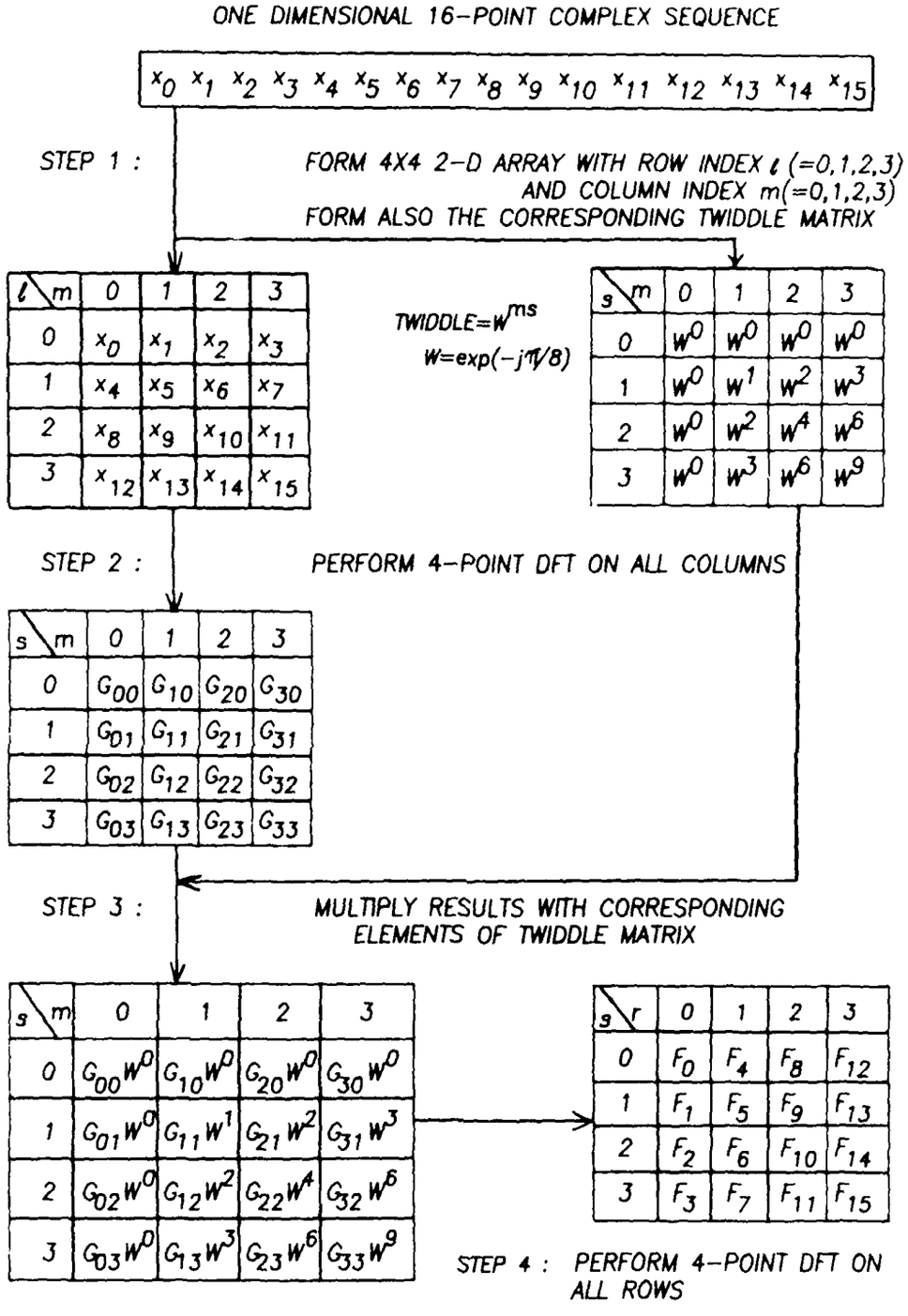
w_3 w_2 w_1 w_0 z_3 z_2 z_1 z_0 y_3 y_2 y_1 y_0 x_3 x_2 x_1 x_0

w_7 w_6 w_5 w_4 z_7 z_6 z_5 z_4 y_7 y_6 y_5 y_4 x_7 x_6 x_5 x_4

w_{11} w_{10} w_9 w_8 z_{11} z_{10} z_9 z_8 y_{11} y_{10} y_9 y_8 x_{11} x_{10} x_9 x_8

w_{15} w_{14} w_{13} w_{12} z_{15} z_{14} z_{13} z_{12} y_{15} y_{14} y_{13} y_{12} x_{15} x_{14} x_{13} x_{12}

FIGURE 16. REQUIRED DATA PERMUATATION FOR A RADIX-4 COMMUTATOR.



ℓ \ m	0	1	2	3
0	x_0	x_1	x_2	x_3
1	x_4	x_5	x_6	x_7
2	x_8	x_9	x_{10}	x_{11}
3	x_{12}	x_{13}	x_{14}	x_{15}

s \ m	0	1	2	3
0	W^0	W^0	W^0	W^0
1	W^0	W^1	W^2	W^3
2	W^0	W^2	W^4	W^6
3	W^0	W^3	W^6	W^9

s \ m	0	1	2	3
0	G_{00}	G_{10}	G_{20}	G_{30}
1	G_{01}	G_{11}	G_{21}	G_{31}
2	G_{02}	G_{12}	G_{22}	G_{32}
3	G_{03}	G_{13}	G_{23}	G_{33}

s \ m	0	1	2	3
0	$G_{00}W^0$	$G_{10}W^0$	$G_{20}W^0$	$G_{30}W^0$
1	$G_{01}W^0$	$G_{11}W^1$	$G_{21}W^2$	$G_{31}W^3$
2	$G_{02}W^0$	$G_{12}W^2$	$G_{22}W^4$	$G_{32}W^6$
3	$G_{03}W^0$	$G_{13}W^3$	$G_{23}W^6$	$G_{33}W^9$

s \ r	0	1	2	3
0	F_0	F_4	F_8	F_{12}
1	F_1	F_5	F_9	F_{13}
2	F_2	F_6	F_{10}	F_{14}
3	F_3	F_7	F_{11}	F_{15}

FIGURE 17. GENERALIZED FFT PROCEDURES APPLIED TO THE RADIX-4 SIXTEEN-POINT CASE.

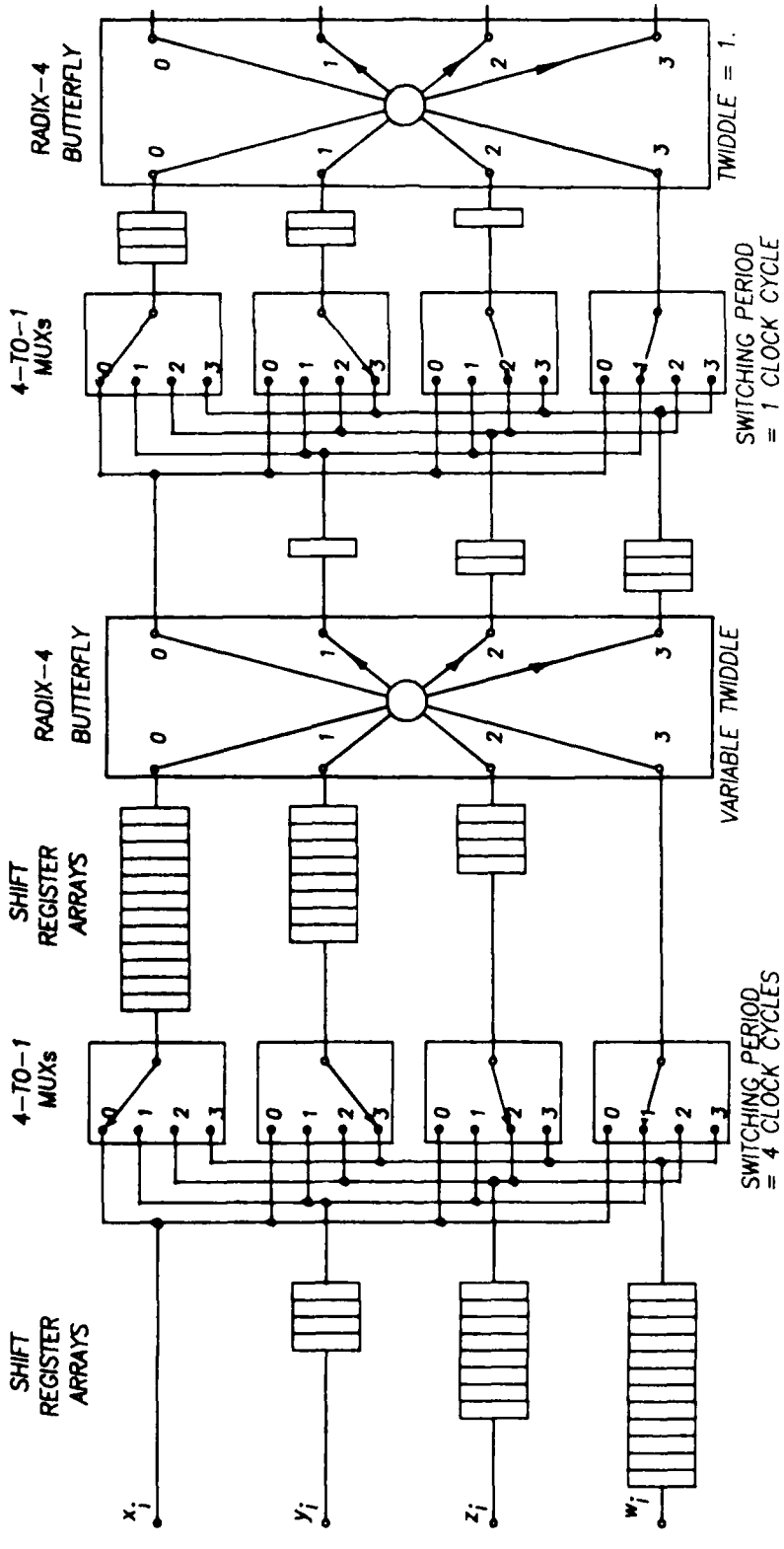


FIGURE 18. SCHEMATIC DIAGRAM OF A SINGLE PIPELINE RADIX-4 SIXTEEN-POINT FFT.

Table XV: Composite component counts of the single pipeline radix-4 16-point FFT.

Type of component	quantity
real adders	38
real multipliers	12
latches	86(including input latches)
4-to-1 multiplexers	8

In the last two sections, several pipeline FFT structures were illustrated based on a 16-point FFT example. The composite component counts for the various structures are compared in Table XVI for N=16.

Table XVI: Comparison of component counts for various 16-point pipeline FFT structures

Component	Parallel-pipeline		Single-pipeline	
	radix-2	radix-4	radix-2	radix-4
real adders	148	144	22	38
real multipliers	40	32	12	12
latches	308	240	78	86
2-to-1 MUXs	-	-	12	-
4-to-1 MUXs	-	-	-	8

If the length of the data sequence is moderately large, the component counts for the parallel pipeline structure become prohibitively large. As an example, the component counts for various pipeline structures for a 64-point FFT are summarized in Table XVII.

Table XVII: Composite component counts for various pipeline structure of the 64-point FFT

Component	Parallel-pipeline		Single-pipeline	
	radix-2	radix-4	radix-2	radix-4
real adders	964	920	34	60
real multipliers	392	304	20	24
latches	2116	1560	202	220
2-to-1 MUXs	-	-	20	-
4-to-1 MUXs	-	-	-	16

It is reasonable to conclude that, using discrete IC elements, the only practical hardware FFT structure is the single pipeline structure for long (longer than 64) data sequences. It is, therefore, useful to derive a generalized single pipeline FFT structure so that FFTs of long data sequences can be processed with varying degree of parallelism.

4.3 Radix-r single-pipeline FFT

Assume in the following discussion that N can be expressed as a power of r. The DFT of an N-point sequence can be efficiently computed using a number of identical sections of processing hardware composed of a radix-r commutator and a radix-r butterfly. The radix-r commutator consists of two sets of shift register arrays with a set of r-to-1 multiplexers sandwiched in between. The schematic diagram of a radix-r pipeline section is shown in Figure 19. Input 'i', where i=0,1,2,..,r-1, of the commutator is connected to input terminal 'i of all the MUXs through a (iN/r)-stage shift register array. The output of the ith MUX is connected to the ith input of the radix-r butterfly through a [(r-i-1)N/r]-stage shift register array. The function of a radix-r commutator is to decimate the input sequence into r sub-sequences of length N/r and present them simultaneously at the output of the commutator. The length of the data sequences reduces by a factor of 1/r and the number of sub-sequences increases by a factor of r, thereby, rendering the total number of samples unchanged at each stage. Consequently, the switching period of the MUXs is given by N/r^k, $k=1,2,3,\ldots, \log_r N$. The switch connections of the MUXs follow a cyclic, pattern, with the ith MUX always one position behind that of the (i-1)th MUX.

The twiddle factors for the radix-r butterfly in each stage of the pipeline are variable and the values must be brought in from memory. The switching period for the twiddle factors in each pipeline section is a

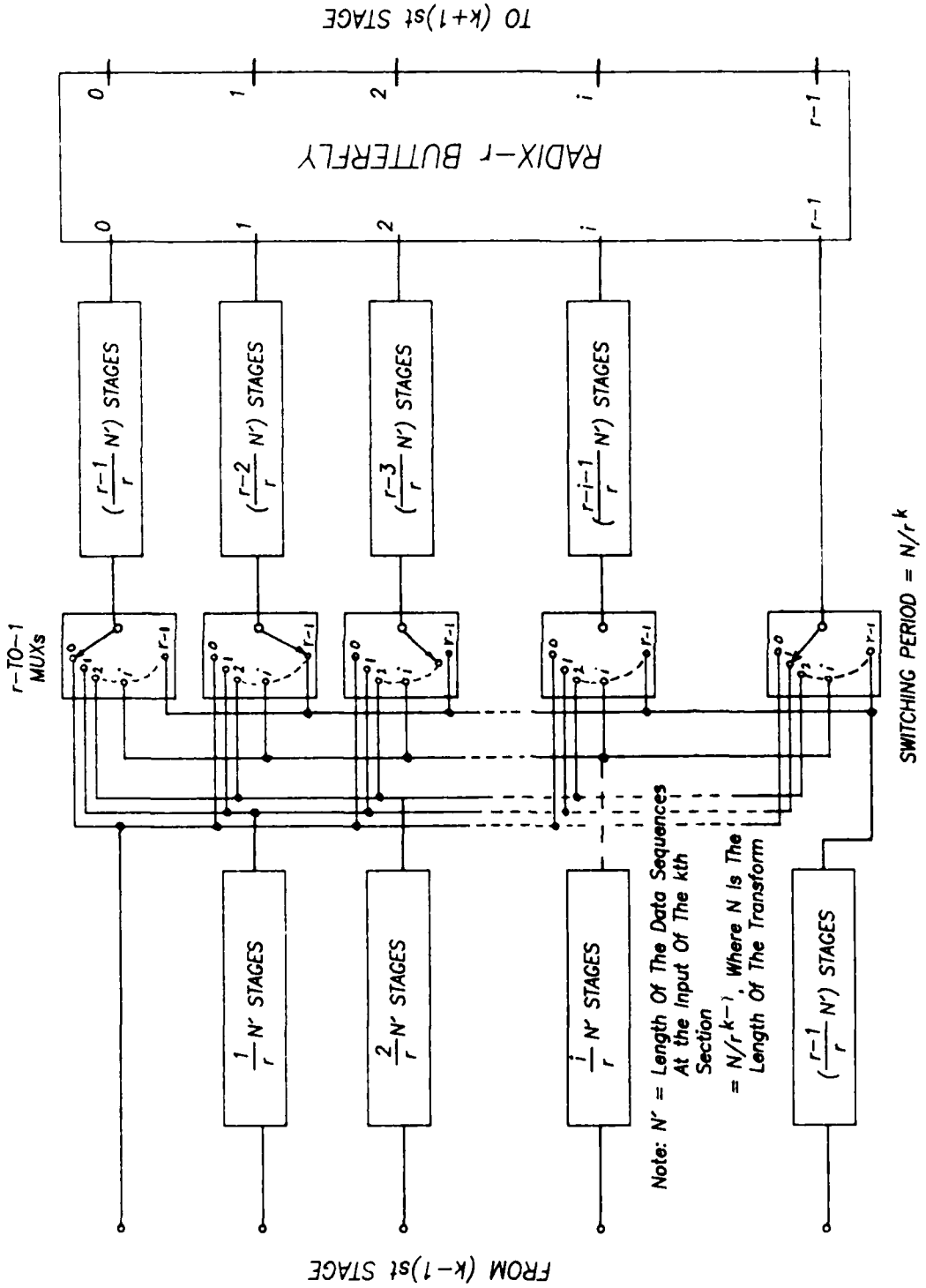


FIGURE 19. THE kTH SECTION OF A RADIX-r SINGLE PIPELINE FFT.

function of the type of decimation process being used. In general, it increases at a rate of r times per section. The direction of increase of the switching period for the twiddle factors depends on whether the DIT or the DIF process is used. For the DIF process, the shortest switching period is at the input stage, in which one set of twiddle is switched in every clock cycle. For the DIT process, the shortest switching period is at the last stage. The twiddle values can be obtained from the appropriate signal flow diagram of the FFT algorithm being implemented.

It can be easily shown that the single pipeline FFT structure corresponding to specific values for r, N and the appropriate choice of decimation process can be derived from the generalized structure given in Figure 19. For example if one substitute r=2, then the length of the shift register arrays at the first stage is N/2 and the r-to-1 MUXs become 2-to-1 MUXs. It can be seen that Figure 19 reduces to Figure 13 which represents the radix-2 single pipeline FFT. The radix-r single pipeline FFT is useful in the design of high speed FFT processors of long data sequences. We can construct the pipeline sections with butterflies of moderately large radix such as 8 or 16. The component counts for these butterflies, although very large, are not prohibitive. Consider, for example, a single pipeline 4096-point FFT. This processor can be implemented with 3 stages of pipeline sections composed of a radix-16 commutator and a radix-16 butterfly. A radix-16 butterfly is implemented by adding a twiddle multiplication stage to a parallel pipeline 16-point FFT such as the one described in Section 4.1. Assume that the pipeline section can handle one set of 16 complex samples in 100 nsec., then it will only take 0.1x4096/16 = 25.6 microsec. to compute the 4096-point FFT.

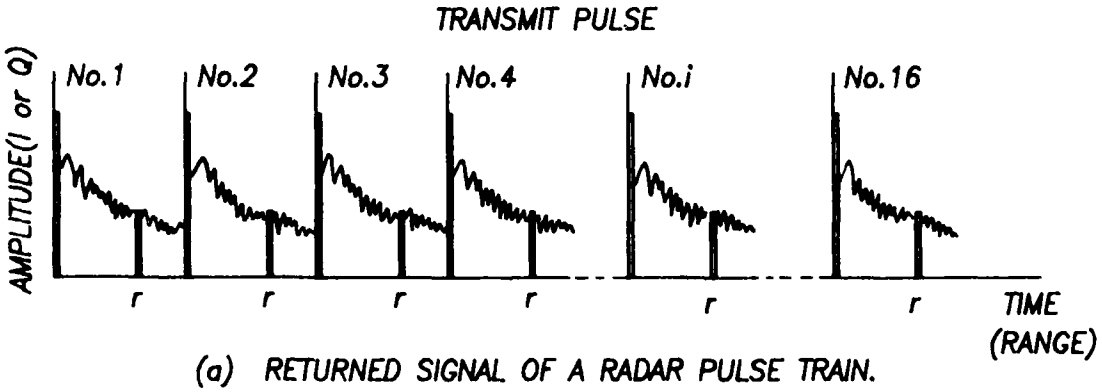
5. DESIGN FOR A DOPPLER PROCESSOR AND A 2-DIMENSIONAL DIGITAL BEAMFORMER BASED ON A HIGH-SPEED 16-POINT FFT IMPLEMENTATION.

In this section is described a high-speed FFT processor of relatively small dimensions. Two applications in radar signal processing are used as examples of possible area of utilization for the processor; these are, (i) Doppler Processing and (ii) two-dimensional digital beamforming.

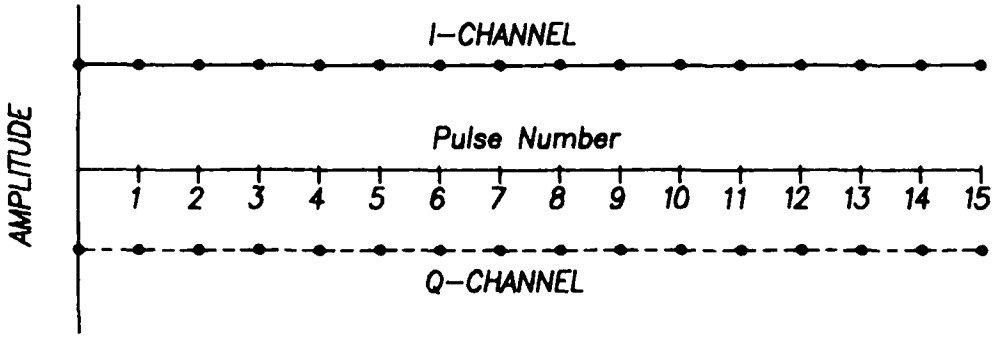
5.1 Doppler processing and digital beamforming

(a) Doppler processing

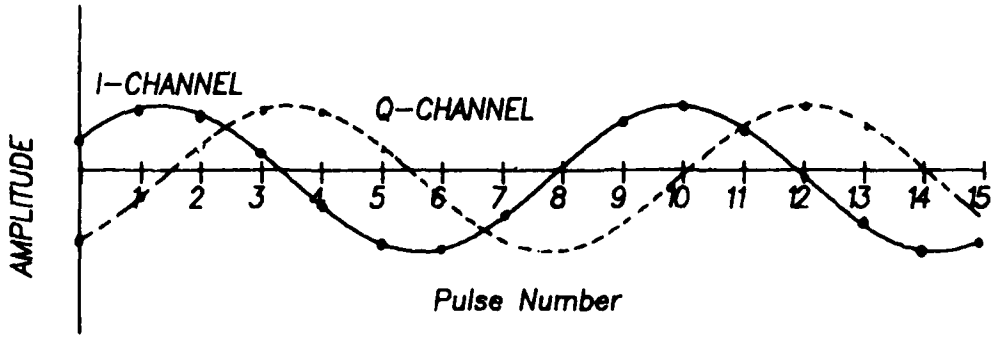
In modern surveillance radar systems, high performance is attained by exhaustively processing the radar signals for all of their available information content. Target velocity, which can be derived from the Doppler shift exhibited by a radar signal, is an important parameter which is used for target identification. It can provide a valuable input for determining the level of threat posed by the target. It can also provide the tracker supplementary tracking information as well as enhancing the detectability of moving targets.



(a) RETURNED SIGNAL OF A RADAR PULSE TRAIN.



(b) IDEALIZED RETURNED RADAR PULSE TRAIN FROM A STATIONARY OBJECT.



(c) IDEALIZED RETURNED RADAR PUSLE TRAIN FROM A MOVING OBJECT.

FIGURE 20.

As is well known, a coherent radar transmits a train of coherent pulses in a fixed direction in space. Radar returns or echoes are received and sampled as depicted in Figure 20a. Since the range of the target being illuminated is not known before hand, one usually samples all of the radar's range cells. The radial extent of a radar range cell is given by:

$$\Delta R = cT/2$$

where

c = speed of light
T = radar pulse width (32)

Consider the returns from an object within a particular range cell R. If the object is stationary, then the echo will be similar to that shown in Figure 20b. On the other hand, if the object is moving with a velocity v, there will be a change in the phase angle of the returned signal from one pulse to another. Detection of this shift is dependent on coherent quadrature demodulation. The sampled waveform from a moving target will be similar to that shown in Figure 20c. This change in phase angle is reflected as a Doppler frequency shift given by:

$$f_D = 2v/\lambda$$

where

v = velocity of target
λ = radar wavelength
f_D = Target Doppler frequency (33)

The DFT of a finite time sequence may be considered as a bank of correlators correlating the time sequence with a set of sampled complex sinusoidal waveforms (replica signals). The frequencies of these sinusoids are evenly spaced across the Doppler frequency band, which, in this case, is equal to the radar pulse repetition frequency(PRF)[7]. If the target possesses a velocity which matches one of the replica signals, the output of that frequency cell in the correlator bank will be much higher than those of the others. Thus the DFT of a returned radar pulse train from a fixed direction and range effectively channels the signal energy components into separate frequency cells according to their Doppler frequencies. For example, the ground return will be channeled into the zero Doppler frequency cell, while the signal component due to a fast moving target is channeled into one or more of the higher Doppler frequency cells.

There are several advantages in performing Doppler processing. The first is that strong returns from ground clutter will appear in low Doppler frequency cells. In order to declare the presence of a target, a detection threshold is normally set according to the past information obtained from that particular range cell. Without Doppler processing, the return from stationary ground clutter and moving targets are indistinguishable. Since ground clutter usually has a large magnitude, it is necessary to set the threshold at a level sufficiently high so as to keep the false alarm rate low. High threshold levels tend to reduce the detection probability of weak targets. With Doppler processing, however, the signal component due to a moving target will appear in a different Doppler frequency cell than that of the ground clutter. The ambient signal level in these high Doppler cells,

i.e., the signal level in the absence of target returns, will have a much lower magnitude because they usually consist of receiver noise only. Receiver noise is wide-band and will be distributed equally in all Doppler frequency cells by the DFT. Consequently, the threshold level for Doppler frequency cells other than that of the zero-Doppler one can be set significantly lower, thereby, enhancing the detectability of moving targets while, at the same time, maintaining a low false alarm rate.

The second advantage of Doppler processing is that it provides an estimate of the target velocity. This estimate is useful in a threat analysis of potential targets as well as in defence resource management. Finally, Doppler processing provides additional tracking information to the tracker. Due to the wide range of possible target speeds, there will not be a one-to-one correspondence between target velocity and target Doppler frequency. This is analogous to the aliasing effects commonly referred to in digital signal processing. Nevertheless, this ambiguity can be resolved by employing multiple PRFs [8] in the radar system.

Consider a surveillance radar system employing a phased array antenna such as the one described by Mabey [9]. The main beam of this phased array can be stepped electronically and maintained in one direction indefinitely. The surveillance area of this antenna is ±40 degrees in azimuth and ±30 degrees in elevation. The 3 dB azimuthal and elevation beamwidths of this antenna are 4 and 6 degrees respectively. Assume that a surveillance cell is defined by the beamwidths in both azimuth and elevation. The surveillance area is divided into 126 cells in azimuth and elevation. In order to successfully incorporate Doppler processing in this system, the processor has to be matched to the radar system parameters. Some of the relevant parameters are listed below:

Radar PRF= 1 kHz to 10 kHz
Pulse width = 100 nsec to 400 nsec
Maximum range = 40 km
Dynamic range = 60 dB

The 126 surveillance cells are to be kept under continuous observation within constraints imposed by radar dwell time. Assuming that the refresh rate is one scan per second, the dwell time per surveillance cell per scan would be 1/126 = 7.94 msec. For a typical PRF of 2 kHz, approximately 16 pulses may be transmitted in each direction before the beam is moved to another surveillance cell. If the radar pulse width is taken to be 400 nsec, the number of range cells within the maximum range is approximately 500. Thus a reasonable set of specifications for a Doppler processor for this radar may be determined:

Maximum length of pulse train = 16 pulses
Minimum length of pulse train = 4 pulses
Processing rate = 512 sixteen point FFTs in 2 msec.
wordlength = 12 bit
Data window = Hamming window

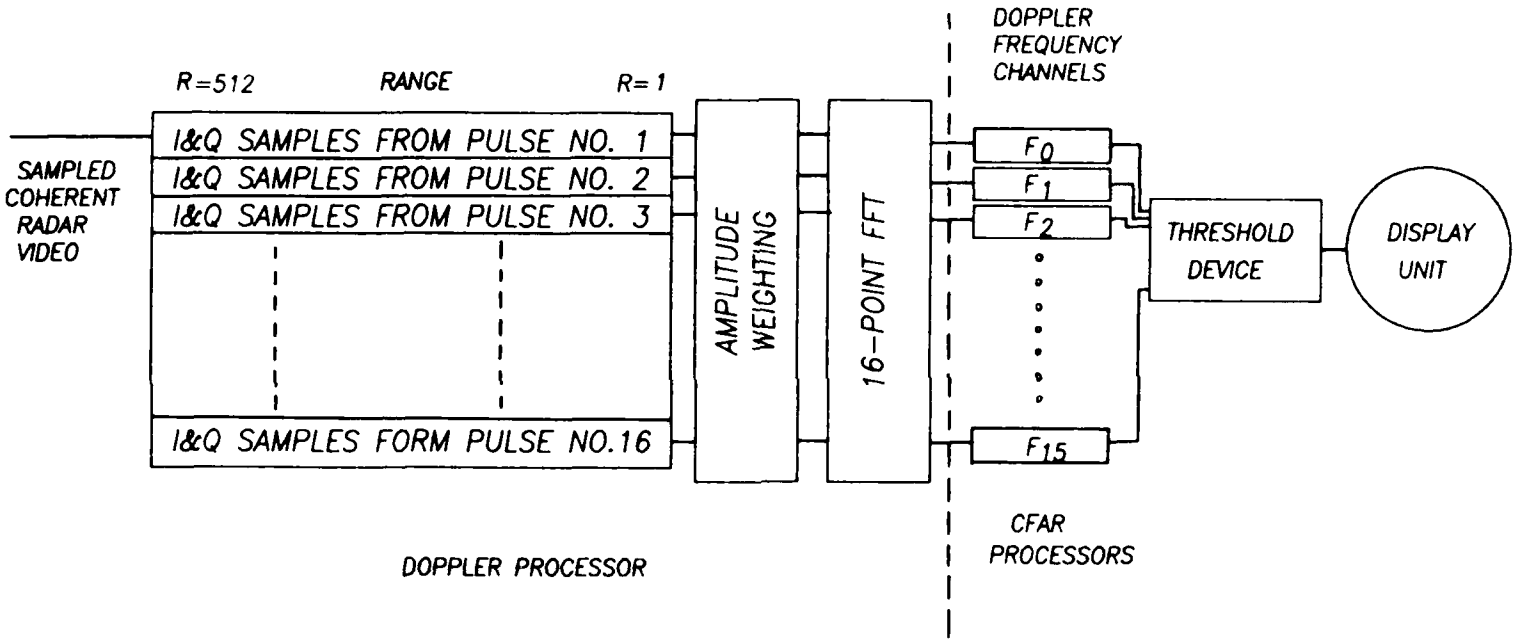


FIGURE 21. BLOCK DIAGRAM OF A RADAR SIGNAL PROCESSING SYSTEM EMPLOYING A DOPPLER PROCESSOR.

Data windows (amplitude weighting of input data) are necessary to suppress the Doppler sidelobes. The DFT of a finite time sequence may be considered as the Fourier Transform of the product of an infinite time sequence with a rectangular pulse function. It is well known [10] that the Fourier transform of the product of two functions is equal to the convolution of the Fourier transforms of the individual functions. Since the Fourier transform of a rectangular pulse is a sinc(sinx/x) function whose first sidelobe has a magnitude only -13.6 dB below that of the mainlobe, the response of a Doppler cell to a strong signal with a Doppler frequency matched to that of a neighbouring Doppler cell would be substantial. This phenomenon is sometimes referred to as leakage in digital signal processing terminology. This may produce ambiguity in the estimate of the target velocity. Ground clutter whose magnitude may be orders of magnitude higher than that of a target signal will cause interference in neighbouring Doppler cells, thereby, degrading detection performance for moving targets. The data windows commonly employed have significantly lower values at the beginning and the end of the time sequence. Thus the data window effectively imposes a much smoother transition than the rectangular window function. The smooth transition results in much lower sidelobes in the DFT of the modified time sequence. The price one pays for applying amplitude weighting is that the width of the DFT main lobe becomes broader than in the case of a rectangular window. However, this is more than compensated for by the resulting low Doppler sidelobes of the Doppler processor. An example of a data window is the Hamming window which has a -42 dB first sidelobe.

A typical block diagram for a radar signal processing system employing a Doppler Processor is shown in Figure 21. For the following discussion, we shall call the samples of the radar returns from all the range cells in a fixed direction a sweep. The coherent video signal is sampled and stored in a memory bank which is capable of storing up to two sets of 16 sweeps of radar returns containing 512 range samples. Once the desired number of sweeps has been obtained, a signal is sent to the Doppler processor to initiate the parallel readout of successive sets of N(where N is the length of the pulse train) samples from each of the 512 range cells. The N samples are than multiplied by a set of amplitude weights before being fed to a 16-point FFT processor. The 16 Doppler frequency outputs are fed to separate Constant False Alarm Rate (CFAR) [11] processors. In each CFAR unit, the magnitude of the output in each Doppler cell is compared against a threshold level for detection. In addition, this output is used to update the threshold setting for that particular Doppler cell. While one set of data are being processed, new data corresponding to the next look direction are written into the other memory bank.

The processing speed requirements for the Doppler Processor are easily fulfilled and can be realized with any one of the designs described in Section 4. However, as will be seen in the next section, the 16-point FFT is also employed in the two-dimensional digital beamformer example and the processing speed demands here are much higher. The philosophy being followed here is one which considers it to be cost effective to develop a common processor structure which can be used in both of these applications. Consequently, the specifications of the basic high-speed 16-point FFT unit

will be determined by the requirements of the 2-dimensional digital beamformer.

(b) Digital beamformer in a sampled aperture radar system

Modern surveillance and tracking radars face many threats to their security. Anti-radiation missiles (ARM) seek out the direction of a radar emission and home in on the radar. Fire control radars must be capable of tracking targets in many directions within a short time span. Consequently, the amount of time a tracking radar can allocate to the tracking of each target is restricted. It is, therefore, envisaged that future radars would have the capability of surveying and tracking many directions simultaneously. This leads to the concept of sampled aperture radar (SAMPAR) [12].

In simple terms, a SAMPAR employs a phased array antenna as sensors and relies on digital signal processing to synthesize the information from various directions. Consider the scenario in which a bistatic radar is illuminating its surrounding hemisphere with a train of coherent pulses through an omni-directional antenna. The returns from all directions are received by a planar phased array. Each sensor on the phased array aperture has an independent coherent receiver. The receivers convert the signal from each sensor down to complex baseband and sample at a suitable rate, thereby, preserving the amplitude and phase relationships among signals from different sensors. Once the data are in the digital domain, they can be used to compute the response of the receiving antenna in many directions. This digital beamforming process requires a very high speed digital processor.

A feasibility study of such a SAMPAR system has been initiated by Litva [12]. An experimental SAMPAR system is to be built based on a square array aperture with 8 rows and 8 columns of receiving elements. Initially the sampled data would have a total bandwidth of 64 x BW, where BW is the bandwidth of the receiver channels. If each receiver has a bandwidth of 2 MHz, the total signal bandwidth will be 128 MHz. In order to reduce the data rate to a more manageable level, the data will be integrated in some fashion before being fed to a digital beamformer. Preliminary specifications require that a complete 2-dimensional beamforming frame to be completed in 4 micro-seconds.

The 2-dimensional beamforming operation can be accomplished using a 2-dimensional FFT processor. The system block diagram of an experimental digital beamformer is shown in Figure 22. The input data are represented as an 8 by 8 matrix as depicted in Figure 23a. The data matrix is augmented with 8 rows and 8 columns of zeros to form a 16 by 16 matrix as depicted in Figure 23b. The data in the augmented matrix are fed in parallel, one column at a time, to a 16-point FFT processor. The transformed results are stored in memory as a matrix (see Figure 23c). After all columns have been processed, the results are then read out row-wise and fed to another 16-point FFT which provides the required 2-Dimensional digital beamforming.

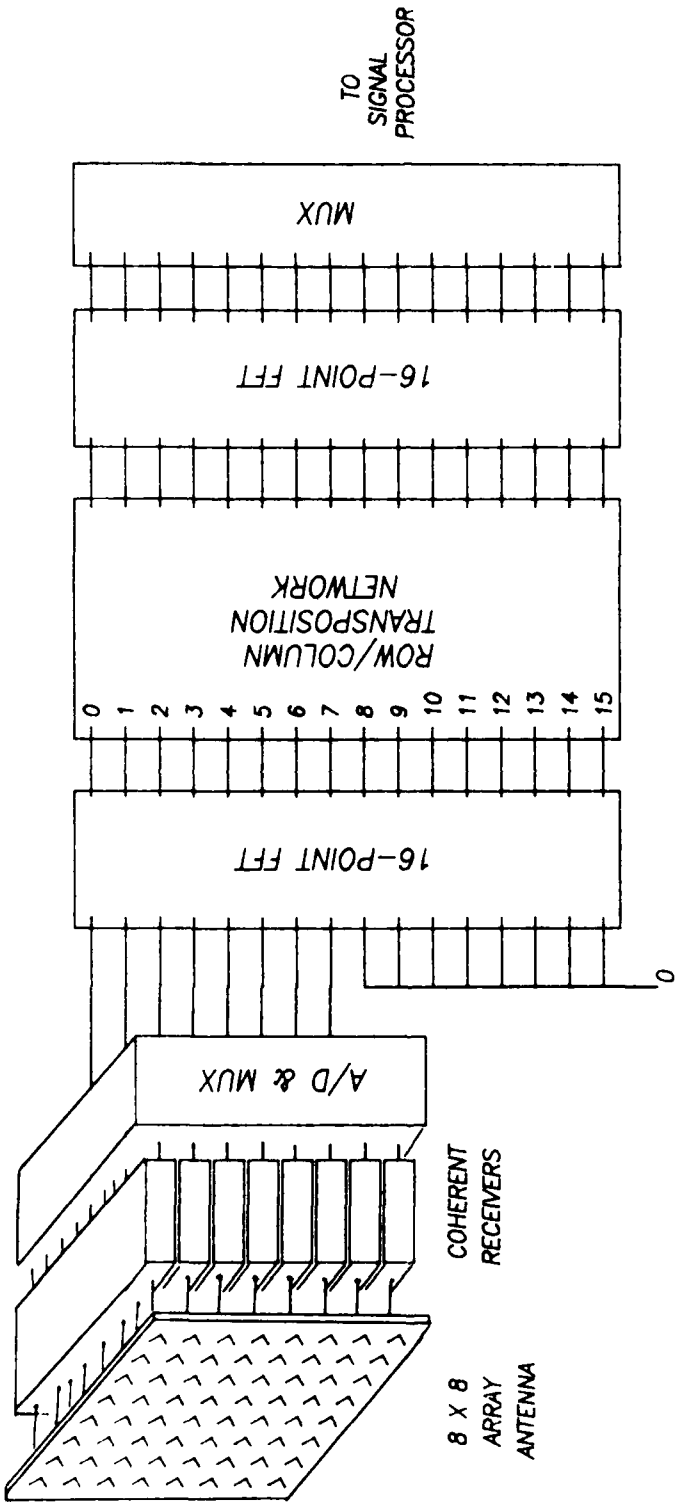


FIGURE 22. SYSTEM BLOCK DIAGRAM OF AN EXPERIMENTAL DIGITAL BEAMFORMER.

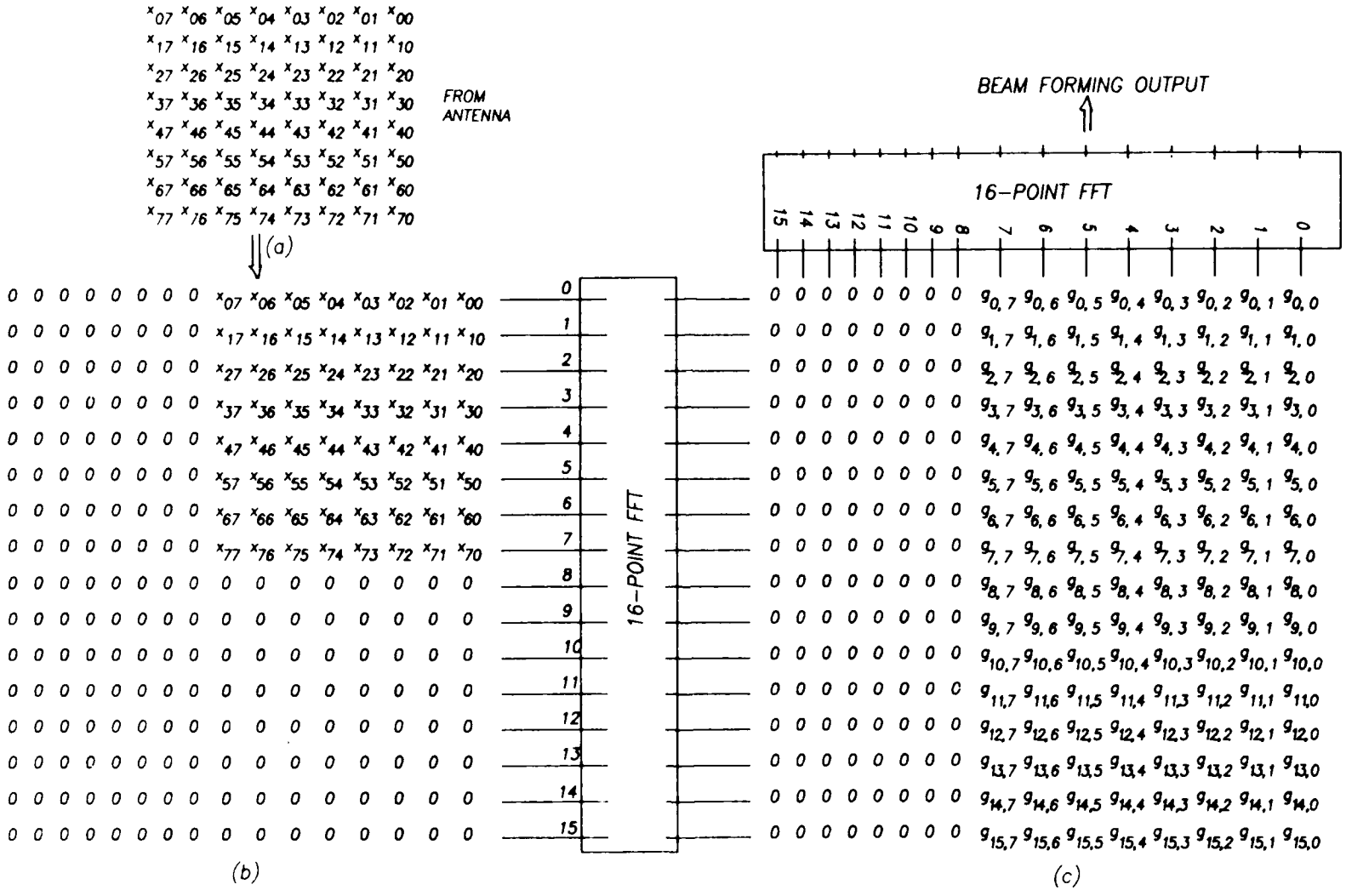


FIGURE 23. DATA FLOW OF THE 2-D BEAMFORMER.

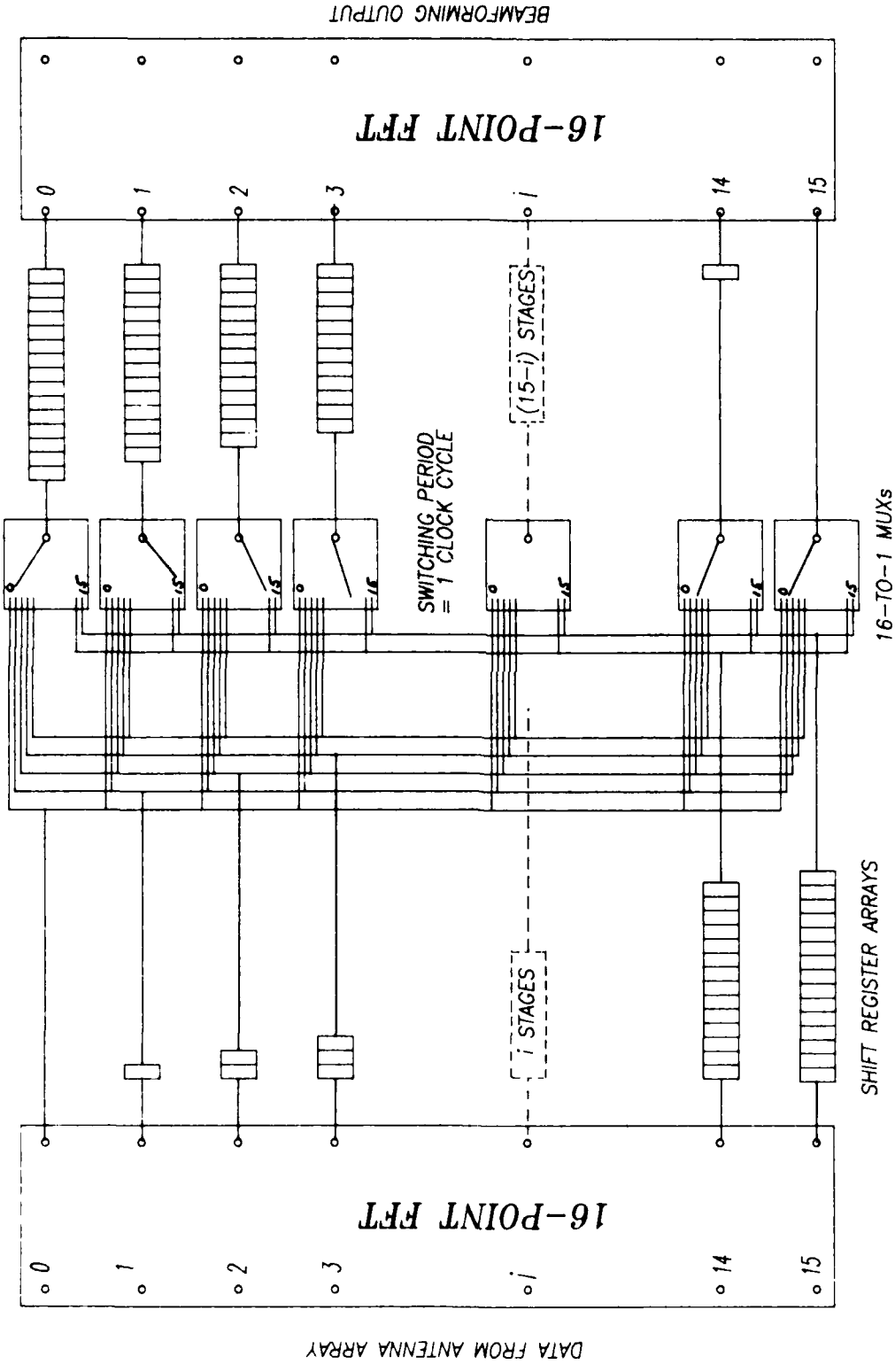


FIGURE 24. SCHEMATIC DIAGRAM OF A 16 X 16 TWO-D BEAMFORMER.

The augmentation of the original 8 by 8 matrix with zeros is a simple way of providing interpolation of the resulting antenna pattern. It effectively provides 16 x 16 = 256 beams in space. Since there are 8 nonzero columns, the first FFT processor in Figure 22 needs only perform eight 16-point FFTs. A 16-point FFT, in general, has 16 non-zero outputs. Thus the second stage FFT would have to perform sixteen 16-point FFTs, although each of them still consists of eight non-zero samples. Presumably, all the 16-point FFTs can be performed by a single 16-point FFT processor. However, this processor must be able to perform twenty four 16-point FFTs in 4 microseconds (or one per 166.7 nsec). Adding all the overhead in data handling, it represents a very high processing speed requirement.

5.2 Implementation of a 2-D digital beamformer and a target Doppler discriminator

After some thought it was decided that the implementation of the system in Figure 22 using inexpensive commercially-available ICs would best be carried out using the parallel pipeline structure described in Section 4.1.1. The schematic diagram of a 2-dimensional digital beamformer employing two separate 16-point FFTs and a row-column transposition network is shown in Figure 24. The data arrangement as indicated in Figure 22 requires a 16 by 16 row-column transposition network. This network requires less hardware than a direct implementation using individually addressable latches. However, the component count is still rather high. Depending on the system data wordlength, the actual IC chip count for this system may be several thousand. Large component counts mean high power consumption which, in turn, present problems with regards to heat dissipation and maintenance.

5.2.1 Modified 2-Dimensional digital beamformer

In an effort to reduce the component count, one must examine the data structure carefully and exploit any peculiar characteristics of the architecture which can be taken advantage of to realize savings in computational effort. Two points come to mind, these are:

(i) The input to both 16-point FFTs has only eight nonzero samples.

(ii) The processing speed requirements for the Doppler Processor is rather moderate. It follows therefore that in both cases, it is feasible to employed an 8-point FFT in multiplexed mode to implement the required 16-point FFT.

(a) modified parallel pipeline 16-point FFT

Let us consider the 16-point DIF FFT algorithm whose operations are summarized in Figure 12. The input data to be transformed are arranged in an 8-column by 2-row matrix. Since for our 2-Dimensional digital beamformer, samples $x_8, x_9, \ldots, x_{15}$ are always zero, the eight 2-point column DFTs may be eliminated. Since two separate 16-point FFTs are required to implement a 2-D FFT, the maximum processing rate for the beamformer is set by the second FFT unit; i.e., to meet our requirements it must be capable of

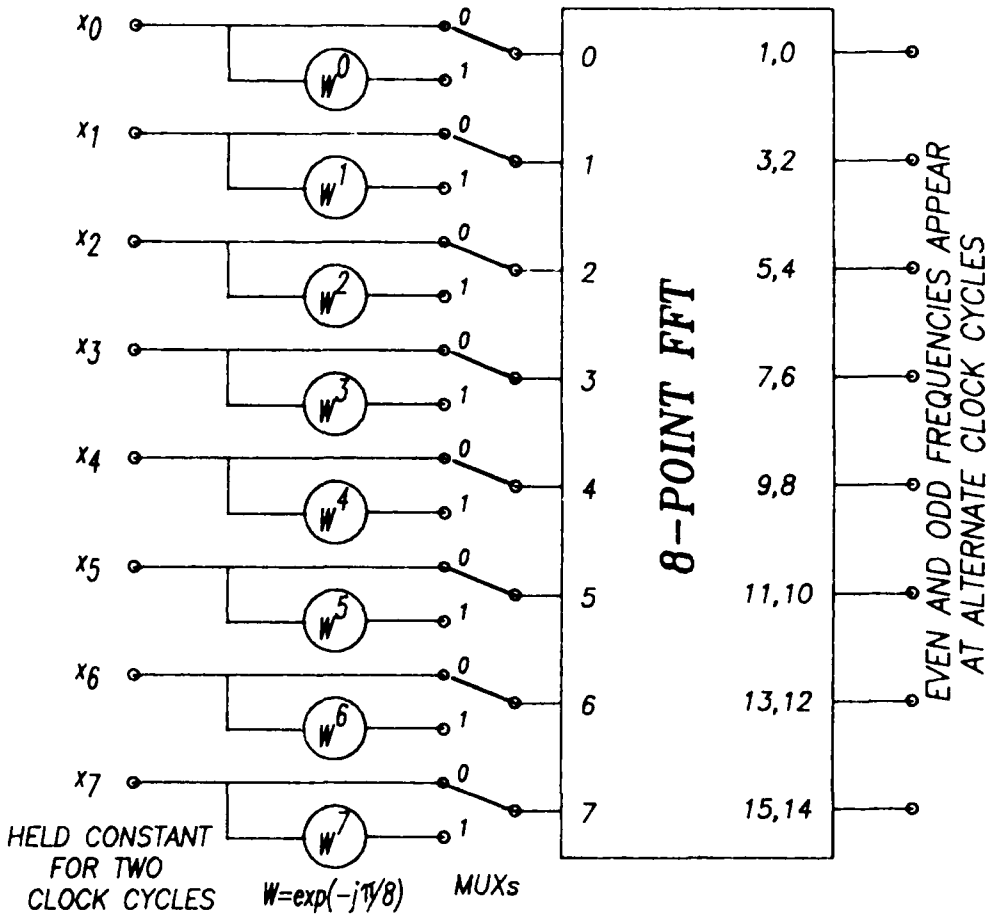


FIGURE 25. SCHEMATIC DIAGRAM OF THE MODIFIED 16-POINT FFT FOR DATA SEQUENCES HAVING ONLY 8 NON-ZERO SAMPLES.

performing sixteen 16-point FFTs per 4 micro-seconds, or one 16-point FFT per 250 nsec. In section 4, it was determined that a conservative estimate of the throughput rate for a parallel pipeline FFT structure is about 100 nsec per FFT. Thus a single 8-point parallel pipeline FFT which is multiplexed to carry out two 8-point FFTs should be capable of completing a 16-point FFT within 250 nsec.

The schematic diagram of a modified parallel pipeline 16-point FFT is shown in Figure 25. This processor takes advantage of the fact that the input has only eight non-zero samples and utilizes an 8-point FFT. The operation of this unit is as follows. Input samples $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7$ are fed to the input of the 8-point FFT in natural order from top to bottom. The input samples are held constant by a set of latches for two consecutive clock cycles. During the first clock cycle, the MUXs are switched to position '0', connecting the input directly to the 8-point FFT module. The result of this 8-point FFT will provide the even frequency outputs (corresponding to the 8-point FFT of the top row in Figure 12). Concurrently, the eight samples are fed to a set of twiddle multipliers which provides the bottom row data in Figure 12. In the second clock cycle, the twiddle multiplication results are ready. The MUXs are switched to position '1', effectively connecting the bottom row of data in Figure 12 to the input of the 8-point FFT. The result of this 8-point FFT provides the odd frequency samples. It should be understood that latches would be inserted in the signal paths to insure proper pipeline operations. Thus the 16-point FFT is obtained in two clock cycles, with even and odd frequency samples appearing at the output of the 8-point FFT processor in alternate clock cycles.

The composite component counts for the modified parallel pipeline 16-point FFT can be estimated. It consists of one eight point parallel pipeline FFT, a twiddle multiplying stage and a bank of 2-to-1 multiplexers. The 8-point FFT has 12 2-point DFTs, 6 trivial and 2 nontrivial twiddles. The twiddle multiplying stage consists of two trivial and six nontrivial twiddles. The bank of MUXs has sixteen (for both real and imaginary parts) 2-to-1 MUXs, yielding the component count given in Table XVIII:

Table XVIII: Component counts of the modified 16-point parallel pipeline FFT

component	quantity
real adders	64
real multipliers	32
latches	128
2-to-1 MUXs	16

(b) An 8 x 16 row-column transposition network

Savings to component counts may be realized once again in the design of the matrix transposition network by taking advantage of the fact that the input to the second 16-point FFT also has only eight nonzero samples. The schematic diagram of such a network is shown in Figure 26. Since in a modified parallel pipeline 16-point FFT the even and odd outputs of

the FFT appears in alternate clock cycles, this modified matrix transposition network also has eight complex inputs and outputs. We shall number them from top to bottom with indices 0 through 7. The data from the first FFT processor is fed to a bank of 8-to-1 MUXs through shift register arrays of various lengths. The length of the shift register array in data line 'i' is given by 2i. For example, there is no shift registers between output '0' of the first FFT and input '0' of all the MUXs (2x0=0). The number of shift registers in the shift register array between output '7' of the first FFT and the MUXs is 14(2x7=14). The output of each shift register array is connected to the corresponding input of all the MUXs. For examples, the output of the shift register array in signalling '1' is connected to input terminal '1' of all 8 MUXs. The output of the ith MUX is connected to the ith input of the second FFT unit through a 2 (7-i) stage shift register array. The MUXs are switched once every two clock cycles in a cyclic pattern, with the output switch position of the ith MUX being always one behind that of the (i-1)th MUX. In Table XIX are tabulated the switch positions for the MUXs through 20 clock cycles.

Table XIX: Switch positions of the MUXs in the 8x16 row-column transposition network over 16 clock cycles

MUX	Input Selected by MUX in Clock Cycle No.									
No.	0,1	2,3	4,5	6,7	8,9	10,11	12,13	14,15	16,17	18,19
0	0	1	2	3	4	5	6	7	0	1
1	7	0	1	2	3	4	5	6	7	0
2	6	7	0	1	2	3	4	5	6	7
3	5	6	7	0	1	2	3	4	5	6
4	4	5	6	7	0	1	2	3	4	5
5	3	4	5	6	7	0	1	2	3	4
6	2	3	4	5	6	7	0	1	2	3
7	1	2	3	4	5	6	7	0	1	2

The operation of this matrix transposition network can now be described. The required data arrangement is shown in Figure 27. At the beginning of clock cycle '0', the even frequency samples of the first column FFT appears at the input of the matrix transposition network. At this timeMUX '0' is at switch position '0' and in general MUX 'i' is at switch position '8-i'. After two clock cycles, the data will be arranged as shown in Figure 26. The MUXs are switched by one position in a cyclic fashion between positions 0 and 7. After another two clock cycles, the data will be arranged as shown in Figure 28a. Following the switching pattern tabulated in Table XIX, it can easily be verified that at the end of clock cycle '15', the data will be arranged as shown in Figure 28b which is consistent with the transposed data matrix in Figure 27.

5.2.2 Doppler Processor employing the modified parallel-pipeline 16-point FFT

The modified parallel pipeline 16-point FFT can also be used to form a Doppler processor with the addition of a multiplexing network. The schematic diagram of this processor is shown in Figure 29. The first stage

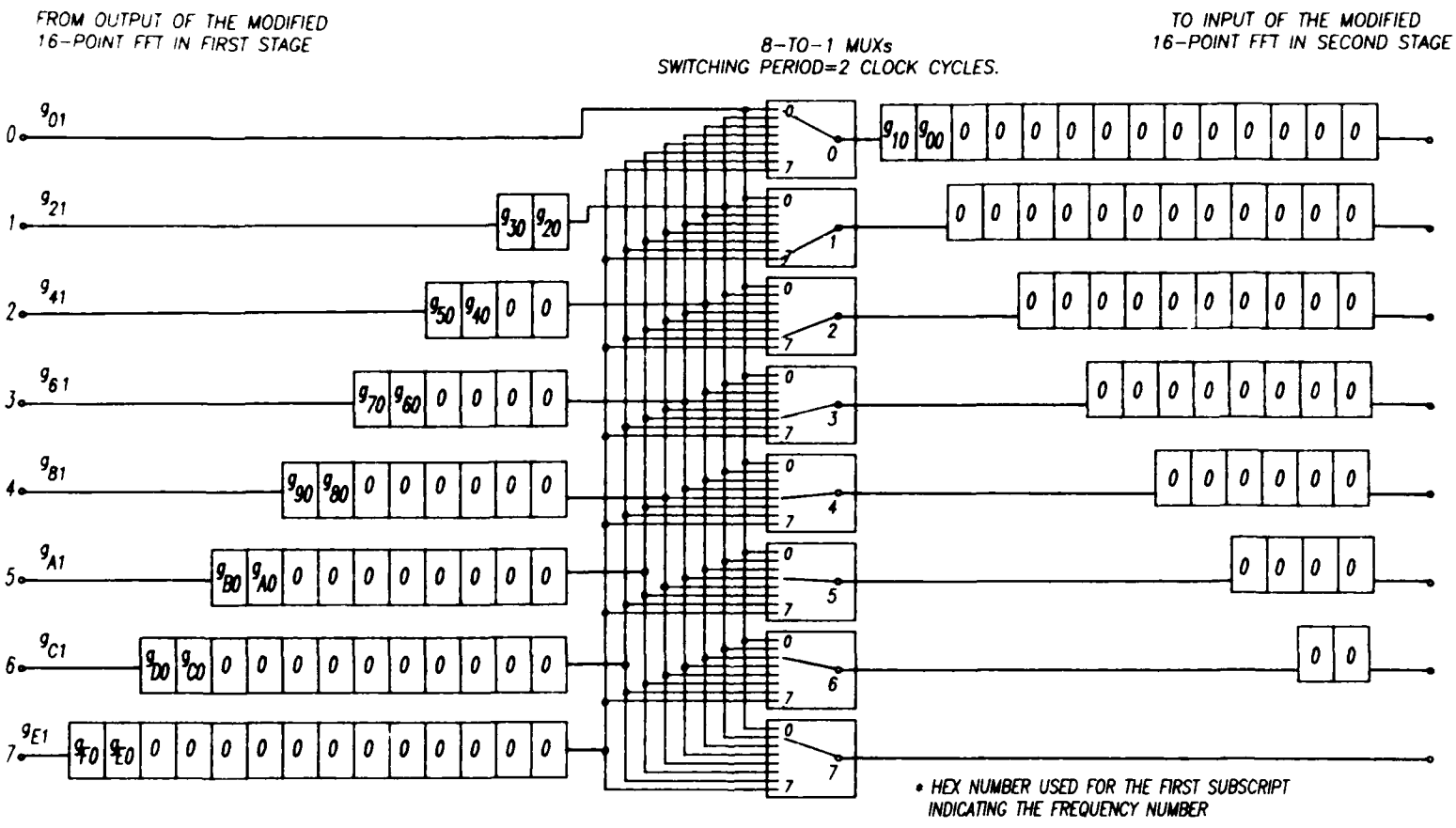


FIGURE 26. SCHEMATIC DIAGRAM OF AN 8 BY 16 MATRIX TRANSPOSITION NETWORK.

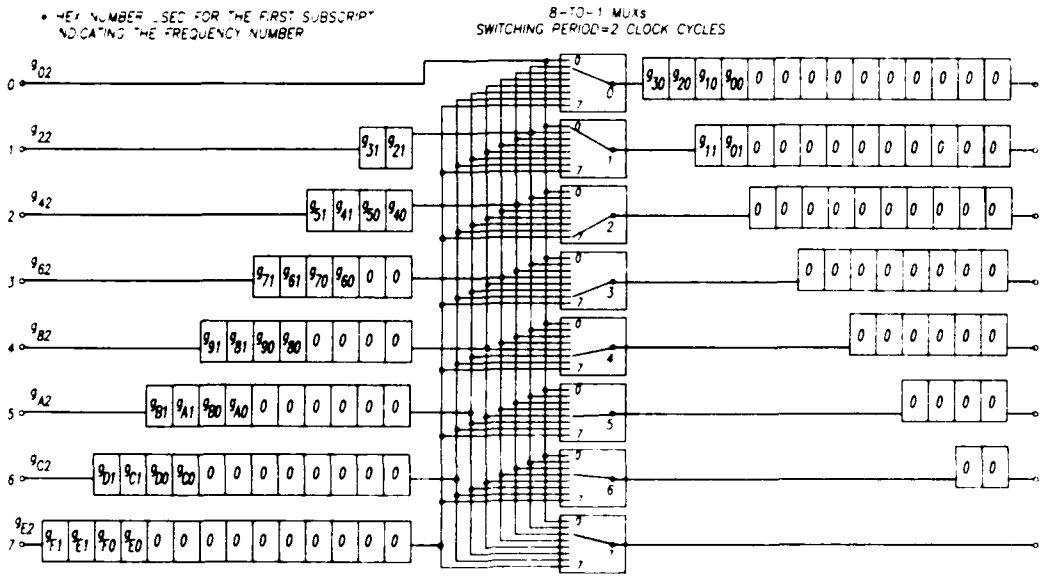
$$
\begin{array}{cccccccccccccccc}
g_{1,7} & g_{0,7} & g_{1,6} & g_{0,6} & g_{1,5} & g_{0,5} & g_{1,4} & g_{0,4} & g_{1,3} & g_{0,3} & g_{1,2} & g_{0,2} & g_{1,1} & g_{0,1} & g_{1,0} & g_{0,0} \\
g_{3,7} & g_{2,7} & g_{3,6} & g_{2,6} & g_{3,5} & g_{2,5} & g_{3,4} & g_{2,4} & g_{3,3} & g_{2,3} & g_{3,2} & g_{2,2} & g_{3,1} & g_{2,1} & g_{3,0} & g_{2,0} \\
g_{5,7} & g_{4,7} & g_{5,6} & g_{4,6} & g_{5,5} & g_{4,5} & g_{5,4} & g_{4,4} & g_{5,3} & g_{4,3} & g_{5,2} & g_{4,2} & g_{5,1} & g_{4,1} & g_{5,0} & g_{4,0} \\
g_{7,7} & g_{6,7} & g_{7,6} & g_{6,6} & g_{7,5} & g_{6,5} & g_{7,4} & g_{6,4} & g_{7,3} & g_{6,3} & g_{7,2} & g_{6,2} & g_{7,1} & g_{6,1} & g_{7,0} & g_{6,0} \\
g_{9,7} & g_{8,7} & g_{9,6} & g_{8,6} & g_{9,5} & g_{8,5} & g_{9,4} & g_{8,4} & g_{9,3} & g_{8,3} & g_{9,2} & g_{8,2} & g_{9,1} & g_{8,1} & g_{9,0} & g_{8,0} \\
g_{11,7} & g_{10,7} & g_{11,6} & g_{10,6} & g_{11,5} & g_{10,5} & g_{11,4} & g_{10,4} & g_{11,3} & g_{10,3} & g_{11,2} & g_{10,2} & g_{11,1} & g_{10,1} & g_{11,0} & g_{10,0} \\
g_{13,7} & g_{12,7} & g_{13,6} & g_{12,6} & g_{13,5} & g_{12,5} & g_{13,4} & g_{12,4} & g_{13,3} & g_{12,3} & g_{13,2} & g_{12,2} & g_{13,1} & g_{12,1} & g_{13,0} & g_{12,0} \\
g_{15,7} & g_{14,7} & g_{15,6} & g_{14,6} & g_{15,5} & g_{14,5} & g_{15,4} & g_{14,4} & g_{15,3} & g_{14,3} & g_{15,2} & g_{14,2} & g_{15,1} & g_{14,1} & g_{15,0} & g_{14,0}
\end{array}
$$

(a) DATA OUTPUT FROM THE MODIFIED PARALLEL PIPELINE 16-POINT FFT, ONE COLUMN PER CLOCK CYCLE.

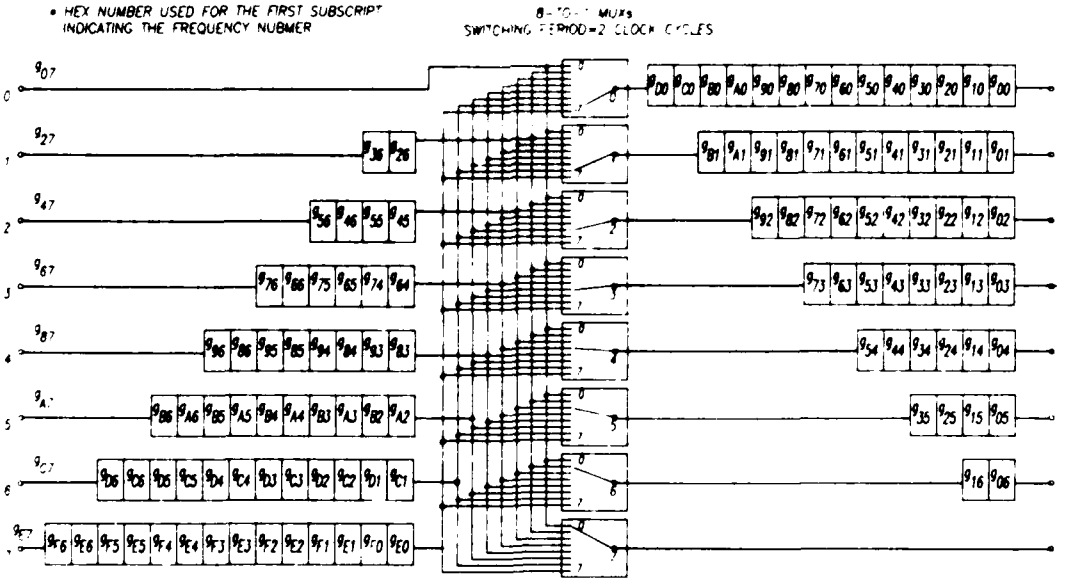
$$
\begin{array}{cccccccccccccccc}
g_{15,0} & g_{14,0} & g_{13,0} & g_{12,0} & g_{11,0} & g_{10,0} & g_{9,0} & g_{8,0} & g_{7,0} & g_{6,0} & g_{5,0} & g_{4,0} & g_{3,0} & g_{2,0} & g_{1,0} & g_{0,0} \\
g_{15,1} & g_{14,1} & g_{13,1} & g_{12,1} & g_{11,1} & g_{10,1} & g_{9,1} & g_{8,1} & g_{7,1} & g_{6,1} & g_{5,1} & g_{4,1} & g_{3,1} & g_{2,1} & g_{1,1} & g_{0,1} \\
g_{15,2} & g_{14,2} & g_{13,2} & g_{12,2} & g_{11,2} & g_{10,2} & g_{9,2} & g_{8,2} & g_{7,2} & g_{6,2} & g_{5,2} & g_{4,2} & g_{3,2} & g_{2,2} & g_{1,2} & g_{0,2} \\
g_{15,3} & g_{14,3} & g_{13,3} & g_{12,3} & g_{11,3} & g_{10,3} & g_{9,3} & g_{8,3} & g_{7,3} & g_{6,3} & g_{5,3} & g_{4,3} & g_{3,3} & g_{2,3} & g_{1,3} & g_{0,3} \\
g_{15,4} & g_{14,4} & g_{13,4} & g_{12,4} & g_{11,4} & g_{10,4} & g_{9,4} & g_{8,4} & g_{7,4} & g_{6,4} & g_{5,4} & g_{4,4} & g_{3,4} & g_{2,4} & g_{1,4} & g_{0,4} \\
g_{15,5} & g_{14,5} & g_{13,5} & g_{12,5} & g_{11,5} & g_{10,5} & g_{9,5} & g_{8,5} & g_{7,5} & g_{6,5} & g_{5,5} & g_{4,5} & g_{3,5} & g_{2,5} & g_{1,5} & g_{0,5} \\
g_{15,6} & g_{14,6} & g_{13,6} & g_{12,6} & g_{11,6} & g_{10,6} & g_{9,6} & g_{8,6} & g_{7,6} & g_{6,6} & g_{5,6} & g_{4,6} & g_{3,6} & g_{2,6} & g_{1,6} & g_{0,6} \\
g_{15,7} & g_{14,7} & g_{13,7} & g_{12,7} & g_{11,7} & g_{10,7} & g_{9,7} & g_{8,7} & g_{7,7} & g_{6,7} & g_{5,7} & g_{4,7} & g_{3,7} & g_{2,7} & g_{1,7} & g_{0,7}
\end{array}
$$

(b) DATA OUTPUT FROM THE 8 X 16 TRANSPOSITION NETWORK, ONE COLUMN PER CLOCK CYCLE.

FIGURE 27.



(a) DATA ARRANGEMENT AND SWITCH POSITIONS AT THE END OF CLOCK CYCLE #3.



(b) DATA ARRANGEMENT AND SWITCH POSITIONS AT THE END OF CLOCK CYCLE #15

Figure 28 Operation of the 8x16 row-column transposition network at various clock cycles

of this Doppler processor using the modified 16-point FFT unit consist of (i) amplitude weights and (ii) adders. Since the input data to the Doppler processor may contain all nonzero values, the adder stage is required to perform the two-point DFTs of the columns as indicated in Figure 12. The input samples are held constant for two clock cycles. In the first clock cycle, the adders perform the difference terms of the 2-point DFTs and pass the results onto the twiddle multipliers. In the second clock cycles, the adders perform the sum terms of the 2-point DFTs, and the results are fed to the 8-point DFT module through a bank of 2-to-1 MUXs. Since the twiddle multiplications involve both multiplication and addition, the results may be made available to the 8-point FFT module after the DFT of the sum terms is completed. This may require inserting some latches in the signal path to provide the pipeline operation. The results are multiplexed so that even and odd frequency samples appear at the output of the 8-point FFT module in alternate clock cycles.

Prototypes of the designs of the Doppler Processor and the Digital Beamformer have been implemented By Interactive Circuits and Systems (ICS) [13-15] for the CRC. These prototypes are shown in Figures 30 and 31, respectively. The FFT units in these two processors employ the modified parallel pipeline design described in Section 5.2. These FFTs are capable of performing one 16-point FFT in 250 nano-seconds. Integer arithmetic is employed, and the data wordlength is 12-bits.

6. HIGH SPEED FFT PROCESSORS FOR OTHER RADAR APPLICATIONS

6.1 Digital pulse compression and radar image processing

In the previous section, the designs of two digital signal processors employing a high speed 16-point FFT were discussed. In both the Doppler processor and the experimental 2-D digital beamformer, an FFT of small dimension (16) is required. These two designs employ integer arithmetic. Because the dimension of these FFTs are small the wordlength required to obtain small quantization errors and large dynamic range are relatively moderate. However, there are other radar applications which require high speed FFT processors of large dimensions. In this section, we shall discuss some of the important applications and explore means of achieving high throughput rates for FFT structures designed for these applications.

(a) Pulse compression matched filter

In modern radars, the transmit pulse is often encoded to provide the radar with high range resolution and while, at the same time, maintaining a high average value for the signal level. These radars are called pulse compression radars [16]. Frequency modulation (FM) is commonly used for implementing pulse compression. The returned signal is processed by a correlator which measures the degree of similarity between the received waveform and the stored replica of an expected waveform. Consequently, this correlator is given the name matched filter. The bandwidth of the FM pulse is in the order of some tens of MHz. In the past matched filters were usually implemented using surface acoustic wave (SAW) devices which offer the advantage of compactness and reasonably good filter characteristics.

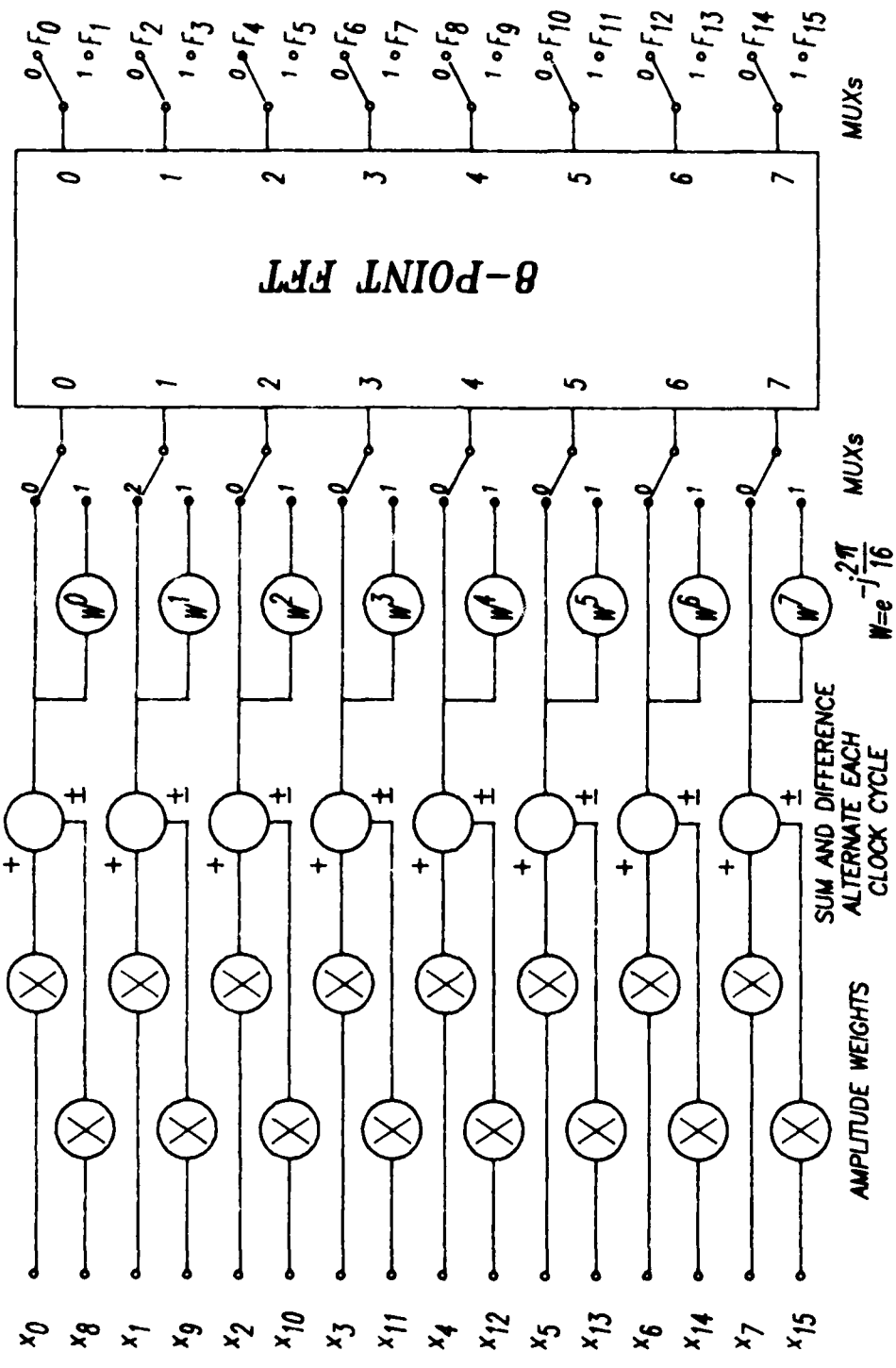


FIGURE 29. SCHEMATIC DIAGRAM OF A 16-BIN DOPPLER PROCESSOR.

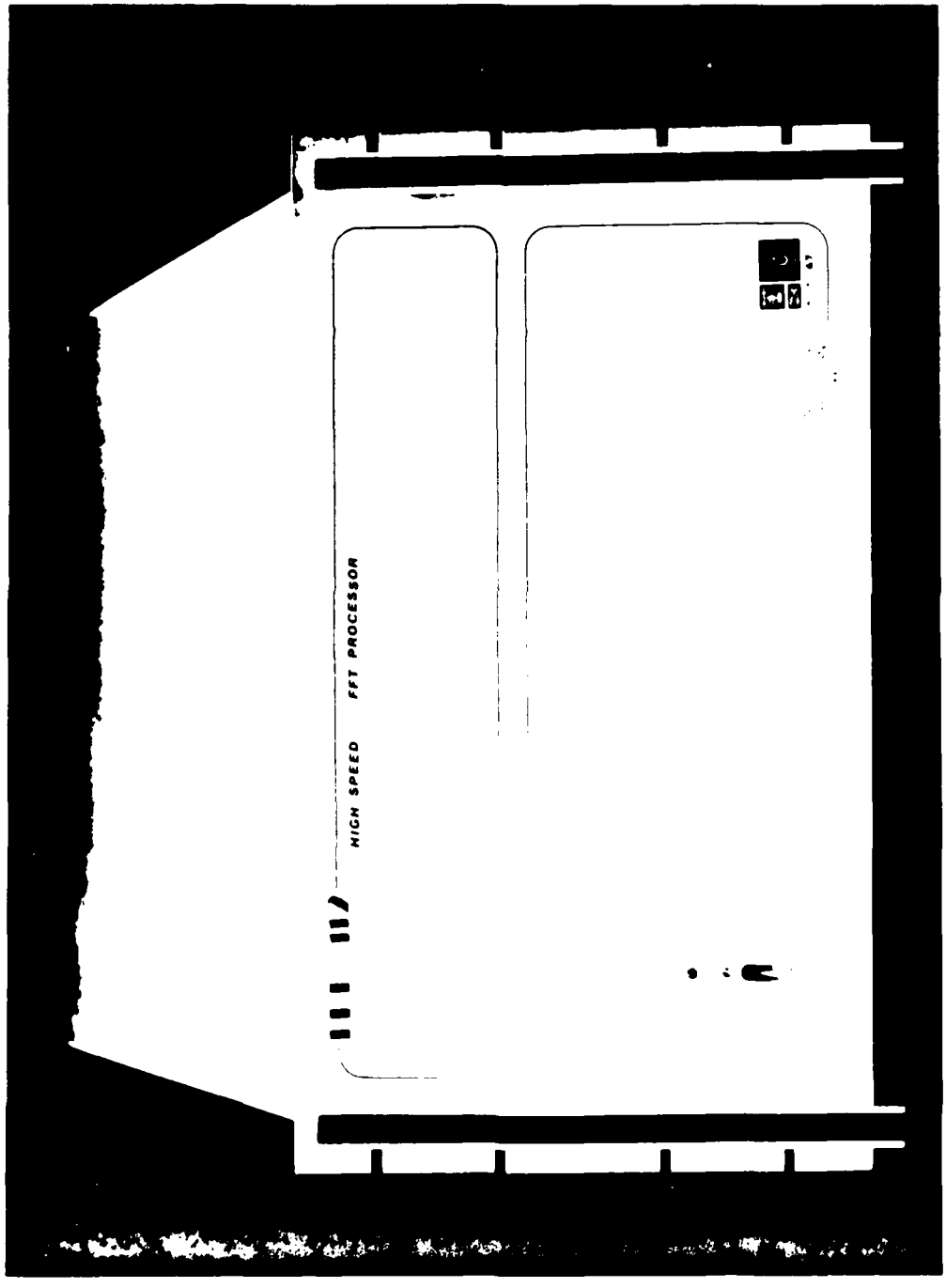


Figure 30 Prototype of the Doppler Processor

Figure 31 Prototype of the 2-D digital Beamformer

For high performance radars, the capability of employing waveforms optimized for particular operational environments is an important feature. Consequently, digital matched filters become very attractive because the filter characteristics can be readily changed. In order to represent the received waveform adequately in the digital domain, it is usually sampled at a rate equal to the signal bandwidth. The entire sampled sequence is divided into sub-sequences of a suitable length, which is usually determined by the time-bandwidth product. A single-pipeline radix-2 FFT of dimension equal to twice the length of the sub-sequences is used to obtain the DFT of two contiguous subsequences at a time. The result of each FFT is then multiplied by the matched filter transfer function. The resulting sequences are then processed using an inverse Fast Fourier transform (IFFT) to give the matched filter output. Except for the order in which the twiddles are applied, the IFFT and the FFT have identical structures. Since the length of the FFT is twice the length of the data sub-sequence, half of the output sequence may be discarded to avoid aliasing effects. Subsequent FFTs use data which overlap the preceding data by one subsequence. This process continues until all returned samples are processed. All these operations must be performed within a pulse repetition interval.

Digital matched filters employing pipeline FFT structure [17] have been implemented successfully for signal bandwidth of 10 to 20 MHz. Recent advances in solid-state technology have increased the feasibility of performing matched filtering digitally on signals with even greater bandwidths. Real-time matched filtering of wide-band waveforms requires high speed FFT processor of large dimensions.

(b) Synthetic Aperture Radar image processing

Synthetic Aperture Radar (SAR) [18]-[21] is a technique which permits an antenna with relatively small aperture size to obtain high resolution radar images both in the azimuthal and range dimensions. A SAR employs pulse compression in range to achieve high range resolution and coherent integration of returns from fixed ranges at different times to achieve high azimuthal resolution. The signal processing in a SAR system may be thought of as a two-dimensional cross-correlation of a set of radar echo data with the response function of a point target. Real-time image processing of SAR data is often done with optical systems because of the extremely high data rate. With ever increasing speeds of digital hardware, near real-time digital SAR processing appears to be feasible in the near future. Digital SAR processors provide some unique advantages over other signal processing systems. For example, the signal processing system may be required to identify signatures of maneuvering targets [21]. A digital SAR processor could provide more accurate phase correction of the data, thereby, allowing a more accurate focusing of the synthetic aperture than possible with an optical processing system.

Digital SAR processing relies heavily on the FFT algorithm. There are a few algorithms [22],[23] which provide more efficient computation of two-dimensional correlations than conventional FFTs. However, the hardware

realization of these algorithms is still FFT-like. Consequently, considerations for hardware implementation of FFTs should apply equally well to these algorithms.

6.2 Component count reduction versus throughput reduction

Two important considerations in the implementation of hardware processors are the component count and the processor throughput. Although it is difficult to obtain an accurate IC chip count of a processor until the actual digital circuit design is chosen, it is possible to get a first order estimate of the chip count from the composite component count and by taking into account certain structural assumptions with regards to each signal processing component. As an example, let us consider the following two cases: (i) a 2-D digital beamformer employing a fully parallel-pipelined 16-point FFT with a 16x16 matrix transposition network, and (ii) one employing the modified parallel-pipeline 16-point FFT and a 8x16 matrix transpositon network. The following assumptions are made with respect to the various composite signal processing components:

(i) real addition is implemented with 4-bit adders
(ii) the latches and shift registers incorporate 8-bits per chip
(iii) multipliers have 12 bits per chip
(iv) 2-to-1 MUXs have 4 bits per chip
4-to-1 MUXs have 2 bits per chip
8-to-1 MUXs have 1 bit per chip
16-to-1 MUXs have 1 bit per chip(larger chip)
(v) data wordlength = 12 bits

The approximate IC chip counts are obtained by multiplying the composite component counts by the estimated number of IC chips required to implement each component and then summing. For example, since 4 bit adders are used, each composite adder will consists of 3 ICs. Similarly, since 8-bit latches are used, each shift register has 1.5 ICs. The corresponding IC chip counts for the two cases are given in Table XX.

Table XX: Comparison of IC chip counts for the two implementations of a 2-D digital beamformer.

IC chip	Fully parallel pipelined 16-point FFT + 16x16 matrix transposition network	Modified parallel pipeline 16-point FFT + 8x16 matrix transposition network
4-bit adders	864	384
multipliers	64	64
latches	1440	720
2-to-1 MUX	-	96
8-to-1 MUX	-	192
16-to-1 MUX	384	-
Total	2752	1456

As can be seen, a reduction of approximately 50% in the IC chip count may be realized by employing the modified design. Other considerations which influence the choice of designs are the physical size of the IC chip and power consumption. For example, a 16-to-1 MUX comes in a 24 pin package, while an 8-to-1 MUX comes in a 16 pin package. The throughput rate of the modified parallel pipeline structure is about half that of the fully parallel pipelined structure. This is expected because, in a parallel pipeline structure, each component is operating at 100% efficiency. Consequently, the throughput will decrease as the amount of hardware is reduced.

Based on the comparison carried out above, it becomes apparent that, if the processing components in a system is operating at 100% efficiency, the only way in which throughput can be increased is by increasing the amount of hardware. Indeed, for a given algorithm, there are only three basic ways to increase the execution speed:

(a) Employ signal processing components (adders, multipliers, etc.) with low intrinsic propagation delays.

(b) Exploit the multiple-stage nature of the algorithm and structure the processor in a pipeline configuration.

(c) Exploit the inherent parallelism in the algorithm and implement parallel branches in the processor.

A number of studies have been carried out with the purpose of examining alternative algorithms for efficient DFT computation. The Winograd Fourier Transform algorithm (WFTA) [24] decreases the number of multiplications relative to the conventional FFT at the expense of increased number of additions. The relative efficiency of hardware implementations of the DFT based on different algorithms cannot be determined easily based solely on a comparison of the number of arithmetic operations. Morris [25] carried out a comparative study of FFT and the WFTA in terms of execution time and storage requirements. He concluded that the WFTA offers neither time nor space advantages over the FFT. However, the comparison is based on the criteria of their relative execution time and their memory requirements amongst a class of general purpose computers. Certain operations which are necessary in a general purpose computer may be eliminated in a hardware implementation. For example, the data shuffling operation may be done by simply rearranging connecting wires in a parallel- pipeline structure. In a technical report, Hicks [26] compared the arithmetic requirements for the prime factor algorithm (PFA) [27], the WFTA, the SWIFT algorithm [28], the DFT and various FFT algorithms. He found that the WFTA was the most efficient algorithm in terms of the required number of real multiplications. The next most efficient algorithm was the PFA. The decision for selecting a particular algorithm from the available alternatives should be based on the following, i.e., one should:

(a) Analyze each algorithm and identify all parallel branches and pipeline stages.

(b) Select a design for each algorithm which offers identical throughput rate.

(c) Define a composite performance criterion based on:
  (i) the hardware component count,
  (ii) type of components,
  (iii) circuit topology(simplicity, regularity, etc.)
  (iv) complexity of control
  (v) overall cost and power consumption.

(d) Select the structure which offers the best performance characteristics.

6.3 Parallel pipeline FFTs of larger dimensions.

Theoretically, one could exploit the parallel pipeline structure to its utmost and realize throughput rates many orders of magnitude higher than the basic system clock frequency. However, there are some fundamental problems which place a practical limit on the size of the FFT which can be implemented in the parallel pipeline structure, using commercially available discrete digital ICs.

The first problem is the dynamic range requirements of the processor. Consider a radix-2 FFT employing integer arithmetics, the result at each stage is obtained by adding two numbers. Consequently the data word size grows at a rate of one bit per stage. In order to maintain the hardware word length, the result at the output of the butterfly must be down-

shifted by one bit. In addition, the twiddle multiplication also introduce round-off errors or truncation errors. These errors will accumulate in successive stages. Hence for large-dimension FFTs, floating point arithmetic is often required. If integer arithmetic is employed, the data-word lengths for the registers tend to be large, and also complicated adaptive scaling techniques need to be used. The whole area of quantization and truncation errors has been treated extensively in the literature [29]-[31]. It has been shown that the quantization effect is significantly lower in a floating point processor than in a fixed point processor.

The second problem is the large number of ICs required as the number of parallel branches increases. Examining Table XX, it is seen that a large number of the required ICs consist of latches. Even if none other than one of the branches in a signal processing pipeline involve processing, latches are required in all parallel branches to ensure synchronized operation. Consequently, the number of latches increases rapidly as the number of parallel branches increases. It is obvious that it would be impractical to implement a fully parallel pipelined FFT using discrete digital ICs when the order of the FFT exceeds the value 64.

The third problem is the magnitude of the physical space required to accommodate all the wired interconnections. Let us consider an FFT of moderate size, say 256. There are 256 complex (or 512 real) signal lines at each stage. Assuming a wordlength of 12 bits, there would be a total of 512x12=6144 wires distributed between the input, the output and intermediate stages, for a fully parallel pipelined structure.

The first two problems can be alleviated somewhat by developing a class of new signal processing ICs. This class of signal processing ICs should have the following characteristics:

(i) Latches would be provided at either the input or the output of a signal processing component such as adders and multipliers.

(ii) All components (adder and multipliers) would be parallel devices in that each device would handle an entire data word. Preferably, they should be able to handle complex arithmetic.

(iii) An IC chip containing a shift register array would be programmable so as to provide variable delays to the signal measured in terms of clock cycles. This would permit a single IC to provide the necessary amount of delay in a branch of a pipeline section regardless of its complexity.

(iv) The basic clock rate of these devices would be high, but the total propagation delay through these devices might be many clock cycles.

There is no ideal solution to the third problem because in order to process all the data in parallel, the data words must be available concurrently to the processors. Furthermore, once all the multiple stage characteristic and the parallelism of the algorithm have been exploited, the only means of increasing throughput rate is to reduce the propagation delay in

the devices. This might require defining a pipeline section at the logical gate level [32]. In other words, a signal processing component such as a floating point multiplier may consist of many pipeline sections. It may take many clock cycles for the device to obtain one set of results. However the throughput rate is only dependent on the propagation delay of a single logical gate. In order to achieve ICs with these characteristics, one would have to look to Very Large Scale Integration (VLSI) technology. For example, if a radix-16 butterfly and a 16x16 matrix transposition network can be fabricated as monolithic devices, then it may be feasible to design parallel pipeline FFTs of larger dimensions.

ACKNOWLEDGEMENTS

Helpful discussions and the reviewing of the manuscript by Dr. John Litva are gratefully acknowledged. This work is supported by the Department of National Defence, Ottawa, Canada under Research and Development Sub Program 33C.

REFERENCES

[1] G.D. Bergland, 'Fast Fourier Transform Hardware Implementations-A Survey', IEEE Trans. on Audio and Electroacoustics, Vol. AU-17, No. 2, June 1969, pp.109-119.

[2] H.T. Kung and C.E. Leiserson, 'Systolic Arrays(for VLSI)', Sparse Matrix Symp. SIAM, 1978, pp.256-282.

[3] L.R. Rabiner and B. Gold, 'Theory and Application of Digital Signal Processing', Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[4] A.V. Oppenheim and R.W. Schafer, 'Digital Signal Processing', Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[5] E.O. Brigham, 'The Fast Fourier Transform', Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1974.

[6] J.W. Cooley, P. Lewis and P.D. Welch, 'The Fast Fourier Transform Algorithm and Its Applications', IBM Corp., Research Paper RC-1743, February 9, 1967.

[7] M.I. Skolnik, 'Introduction to Radar Systems', McGraw-Hill, New York, New York, 1962.

[8] A.W. Rihaczek, 'Modern Radar: Principles and Design Tradeoffs' Part I of a Short Course Presented by MARK Resources Inc., January 1981.

[9] D.J. Mabey, 'An Outline of the DOC/DND Adaptive Radar Research Program', Communications Research Centre Memo, October 1977.

[10] J.B. Thomas, 'An Introduction to Statistical Communication Theory', John Wiley & sons, Inc., New York, New York, 1969.

[11] F.E. Nathanson, 'Radar Design Principles', McGraw-Hill Book Company, New York, New York, 1969.

[12] J. Litva, 'Introduction to Sampled Aperture Radar Technology', International Electrical and Electronics Conference, Proc. Vol. 1, September 26-28, 1984, pp.140-143.

[13] R. Couvillon, P. Menard and D. Roy, "Development of A One-Dimensional FFT Sub-System For Application In Target Doppler Discrimination", Final Report, Interactive Circuits and Systems, January, 1984.

[14] R. Couvillon, P. Menard and D. Roy, "Development of A Radar Signal Processor Demonstration Unit Using The FFT-216 Processor", Final Report, Interactive Circuits and Systems, January, 1984.

[15] C.E. Cook and M. Bernfeld, 'Radar Signals-An Introduction to Theory and Applications', Academic Press, Inc., New York, 1967.

[16] W. Steenaart and N.U. Chowdary, "Real time Radar Signal Processors, Department of Electrical Engineering, University of Ottawa, March 1984.

[17] L.W. Martinson and R.J. Smith, 'Digital Matched Filtering with Pipelined Floating Point Fast Fourier Transforms(FFTs)', IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-23, No. 2, April 1975, pp.222-234.

[18] W.M. Brown and L.J. Porcello, 'An Introduction to Synthetic-Aperture Radar', IEEE Spectrum, September 1969, pp.52-62.

[19] L.J. Cutrona, E.N. Leith, L.J. Porcello and W.E. Vivian, 'On the application of Coherent Optical Processing Techniques to Synthetic-Aperture Radar', Proc. IEEE, Vol 54, No.8, August, 1966, pp.1026-1032.

[20] C.W. Sherwin, J.P. Ruina and R.D. Rawcliffe, 'Some Early Developments in Synthetic Aperture Radar Systems', IRE Trans. on Military Electronics, Vol. MIL-6, April 1962, pp.111-115.

[21] K. Wu and M. Vant, 'Digital SAR Processor Based on the Coherent Sub-Aperture Addition Technique', International Radar Conference, Paris, 1984, pp.425-429.

[22] B. Arambepola, 'Fast Computation of Multidimensional Discrete Fourier Transforms', Proc. of IEE, Vol. 127, Pt. F, No. 1, February 1980.

[23] T.K. Truong, I.S. Reed, R.G. Lipes and C. Wu, 'On the Application of a Fast Polynomial Transform and the Chinese Remainder Theorem to Compute a Two-dimensional Convolution', IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-29, No. 1, February 1981, pp.91-97.

[24] S. Winograd, 'On Computing the Discrete Fourier Transform', Proc. Nat. Acad. Sci(U.S.), Vol. 73, April 1976, pp.1005-1006.

[25] [illegible]P. Morris, 'A Comparative Study of Time Efficient FFT and WFTA Programs for General Purpose Computers', IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 2, April 1978, pp.141-150.

[26] [illegible]ks, 'Comparison of Arithmetic Requirements for the PFA, WFTA, [illegible], MFFT, FFT and DFT algorithms', U.S. Army Missile Command Technical Report [illegible], November 198[illegible].

[27] [illegible] and [illegible] Parks, 'A Prime Factor FFT Algorithm [illegible] High [illegible]', IEEE Trans. Acoustics, Speech and Signal Process[illegible], Vol. ASSP-29, August 198[illegible], pp.[illegible].

[28] [illegible]

[29] [illegible] 'A [illegible] Fast Fourier Transform [illegible] Analysis', IEEE [illegible]

[30] C.J. Weinstein, 'Roundoff Noise in Floating Point Fast Fourier Transform Computation', IEEE Trans. Audio Electroacoustics, Vol. AU-17, September 1969, pp.209-215.

[31] C.J. Weinstein and A.V. Oppenheim, 'A Comparison of Roundoff Noise in Floating point and Fixed point Digital Filter Realizations', Proc. IEEE, Vol. 57, June 1969, pp.1181-1183.

[32] "Trams Final Report", Computing Devices Company, Report A031/FR, Ottawa, 23 August 1983

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA – R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1 ORIGINATING ACTIVITY
Defence Research Establishment Ottawa
Department of National Defence
Ottawa, Ontario, K1A 0Z4

2a. DOCUMENT SECURITY CLASSIFICATION
UNCLASSIFIED

2b GROUP

3 DOCUMENT TITLE
IMPLEMENTATION OF HIGH SPEED FFTs FOR RADAR SIGNAL PROCESSING

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)
Report

5 AUTHOR(S) (Last name, first name, middle initial)
CHAN, Hing C

6 DOCUMENT DATE
August 1985

7a. TOTAL NO OF PAGES
80

7b NO OF REFS
32

8a. PROJECT OR GRANT NO
33C

8b CONTRACT NO

9a ORIGINATOR'S DOCUMENT NUMBER(S)
CRC Report No. 1394

9b. OTHER DOCUMENT NO.(S) (Any other numbers that may be assigned this document)

10 DISTRIBUTION STATEMENT
Unlimited

11 SUPPLEMENTARY NOTES

12. SPONSORING ACTIVITY
DREO

13 ABSTRACT

Discrete Fourier Transform (DFT) has found wide application in radar and sonar systems. With an increased emphasis on digital signal processing, radar systems have requirements for DFTs with ever increasing data rates. The Fast Fourier Transform (FFT) algorithm permits efficient computation of the DFT. In this work, both the fundamentals of FFT algorithms and their implementation are discussed, with emphasis on hardware design. Designs are presented for two radar signal processors employing high speed FFTs, i.e., (1) a high speed Doppler processor and (2) a two-dimensional digital beam-former.

UNCLASSIFIED

Security Classification

KEY WORDS

Radar - FFT - Hardware

INSTRUCTIONS

1 ORIGINATING ACTIVITY Enter the name and address of the organization issuing the document.

2a. DOCUMENT SECURITY CLASSIFICATION Enter the overall security classification of the document including special warning terms whenever applicable.

2b GROUP Enter security reclassification group number. The three groups are defined in Appendix 'M' of the DRB Security Regulations

3. DOCUMENT TITLE Enter the complete document title in all capital letters. Titles in all cases should be unclassified If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital letter abbreviation in parentheses immediately following the title

4 DESCRIPTIVE NOTES Enter the category of document, e.g. technical report, technical note or technical letter. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final Give the inclusive dates when a specific reporting period is covered

5 AUTHOR(S) Enter the name(s) of author(s) as shown on or in the document Enter last name, first name, middle initial If military, show rank. The name of the principal author is an absolute minimum requirement

6 DOCUMENT DATE Enter the date (month, year) of Establishment approval for publication of the document

7a TOTAL NUMBER OF PAGES The total page count should follow normal pagination procedures, i.e. enter the number of pages containing information

7b NUMBER OF REFERENCES Enter the total number of references cited in the document

8a PROJECT OR GRANT NUMBER If appropriate, enter the applicable research and development project or grant number under which the document was written

8b CONTRACT NUMBER If appropriate, enter the applicable number under which the document was written

9a ORIGINATOR'S DOCUMENT NUMBER(S) Enter the official document number by which the document will be identified and controlled by the originating activity. This number must be unique to this document

9b OTHER DOCUMENT NUMBER(S) If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s)

10 DISTRIBUTION STATEMENT Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as

(1) "Qualified requesters may obtain copies of this document from their defence documentation center."

(2) "Announcement and dissemination of this document is not authorized without prior approval from originating activity."

11 SUPPLEMENTARY NOTES Use for additional explanatory notes

12 SPONSORING ACTIVITY Enter the name of the departmental project office or laboratory sponsoring the research and development Include address

13 ABSTRACT Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph, unless the document itself is unclassified, represented as (TS), (S), (C), (R), or (U).

The length of the abstract should be limited to 20 single spaced standard typewritten lines; 7½ inches long

14 KEY WORDS Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.

END

2-87

DTIC